FEDARENA: A UNIFIED AGENT HARNESS FOR FEDERATED LEARNING POISONING RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating robust aggregation in federated learning (FL) requires attacks that adapt to the defense across clients and training rounds. LLM agents offer a way to explore such adaptations, but interpreting their results requires knowing what they observed, what they changed, and which changes actually affected training. We introduce FedArena, a unified harness for controlled, auditable agent-driven poisoning research. FedArena represents attack and defense operations as typed, composable objects, filters observations and interventions by an explicit threat model, and separates local defensive trials from real training rounds. It supports two agent modes, revising attack programs between complete FL runs (L1) and editing client messages between real rounds (L2), and compares both with scripted and human-designed attacks (L0). In an L1 case on CIFAR-10 against Divide-and-Conquer (DnC), a spectral outlier-filtering defense, the agent adds cross-round memory to an attack program; disabling the memory raises development accuracy by 1.98 pp. The frozen program, which also queries a loss-gradient oracle, ends with lower accuracy than the Min-Max attack on 4 of 5 seeds (54.5% vs. 58.0%). In 35 GPT-5.5 episodes on MNIST under Krum, adding local defensive trials raises the rate at which Krum admits controlled updates from 67% to 89%, yet mean final accuracy drops stay at or below 1.34 pp, below a scripted fixed program (3.74 pp) and Min-Max (7.89 pp). Statistics and full-message views reduce token use by 43–45% without increasing damage. Admission and damage thus diverge, and agent-driven adaptive evaluation should report both.

1 INTRODUCTION

Federated learning (FL) trains a shared model through repeated client updates and server aggregation while keeping data local (McMahan et al., 2017). Robust aggregators limit the influence of malicious updates (Blanchard et al., 2017; Yin et al., 2018), yet attacks tailored to their rules can reveal failures missed by simpler constructions (Fang et al., 2020; Shejwalkar & Houmansadr, 2021). Evaluating these defenses therefore requires examining how attacks are constructed and revised in response to the defense, a central concern in adaptive security evaluation (Tramèr et al., 2020; Nasr et al., 2026).

Such adaptation spans clients and training rounds (Xie et al., 2020a; Zhang et al., 2022). Similarity-based defenses push colluding clients to diversify their messages, while history-aware defenses make an attack's behavior across rounds matter (Fung et al., 2020; Karimireddy et al., 2021). Benchmarks such as FLPoison compare existing attacks under fixed configurations (Zhang et al., 2025), but designing a new coordination rule or cross-round memory still requires a researcher to propose, implement, run, and revise it by hand.

LLM agents can propose programs and revise actions from feedback (Yao et al., 2023; Shinn et al., 2023; Toledo et al., 2025), and recent work tests whether they can break adversarial-example defenses (Carlini et al., 2025). FL offers a structured setting for such revisions through client messages, defensive responses, and repeated rounds. Under fixed permissions and budgets, we ask: *(RQ1) how do agents use feedback to revise poisoning strategies, and (RQ2) do their revisions degrade training as much as scripted and human-designed attacks?*

A general-purpose agent attached to an FL simulator can run experiments, but interpreting its revisions requires three poisoning-specific controls. First, permissions: access to other clients' updates changes the threat model, so observations must be filtered. Second, separate clocks: trials against proxy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

messages are weaker evidence than real rounds and must not advance training. Third, execution records must tie each outcome to the revision that produced it.

We introduce FEDARENA, which implements these controls with typed attack and defense objects, permission checks, and separate trial and execution records. It supports *cross-episode strategy evolution* (L1), which revises programs after complete evaluations, and *within-episode interactive adaptation* (L2), which revises client messages and observation timing during training (Figure 1). As L0 references, we run a scripted fixed program and human-designed attacks such as Min-Max in the same settings; the labels index interaction granularity, not capability.

In an L1 case on CIFAR-10 under DnC, the agent adds cross-round memory to an attack program; disabling it raises development accuracy by 1.98 pp, and the frozen program ends with lower accuracy than Min-Max on 4 of 5 seeds (54.49% vs. 57.95%), although these seeds also served as the selection gate and the program has additional gradient access (Section 5.3). In L2 on MNIST under Krum, adding local trials to round feedback raises mean admission, the fraction of rounds in which Krum selects a controlled message, from 67% to 89%. Yet no agent workflow or information view exceeds a 1.34 pp mean final accuracy drop, below a scripted fixed program (3.74 pp) and Min-Max (7.89 pp). Statistics and full-message views mainly cut decisions and token use (by 43–45%). Admission therefore cannot stand in for damage when agents serve as adaptive attackers.

Our contributions are as follows:

- We introduce FEDARENA, a harness that pairs a program interface for cross-episode strategy evolution with typed, composable attack and defense objects (14 attack and 10 defense recipes) for within-episode adaptation, under explicit permissions and separate trial and execution records.
- In 35 GPT-5.5 episodes, we show that feedback and information access change agent interventions, observation timing, and cost, but higher admission does not yield greater damage: the agents remain below scripted and human-designed attacks.
- We demonstrate feedback-guided attack development, showing how cross-episode search produces a reusable attack program and tracing the effect of its structural revision through ablation and repeated evaluation.

2 RELATED WORK

Federated poisoning and robust aggregation. Model poisoning exploits update statistics, directions, and aggregation constraints (Baruch et al., 2019; Xie et al., 2020b; Fang et al., 2020; Shejwalkar & Houmansadr, 2021); backdoor methods explore scaling, distributed triggers, and persistence (Bagdasaryan et al., 2020; Xie et al., 2020a; Zhang et al., 2022). Defenses use geometric and coordinate statistics (Blanchard et al., 2017; Yin et al., 2018; Pillutla et al., 2022), client similarity, and history (Fung et al., 2020; Karimireddy et al., 2021). These operations ground our message, coalition, schedule, and defense-response dimensions as editable research objects.

Federated experimentation and system synthesis. Flower and FedScale support configurable, scalable FL experimentation (Beutel et al., 2020; Lai et al., 2022); FLPoison standardizes poisoning evaluation (Zhang et al., 2025). Helmsman uses agents for FL-system synthesis (Li et al., 2026). FEDARENA organizes poisoning research through explicit search objects, authorized feedback, and two revision granularities.

Automated research and strategy evolution. FunSearch and Eureka refine programs through evaluation (Romera-Paredes et al., 2024; Ma et al., 2024); AIRA-dojo and ShinkaEvolve study research search and program evolution (Toledo et al., 2025; Lange et al., 2026). Our L1 specializes this cycle to complete federated experiments, linking programs, permissions, defensive diagnostics, and reusable artifacts.

Interactive agents and security research. ReAct and Reflexion ground decisions in feedback (Yao et al., 2023; Shinn et al., 2023); GEM and OpenHands supply interaction infrastructure (Liu et al., 2026; Wang et al., 2025). AgentDojo, CyberGym, and AutoAdvExBench instantiate security research

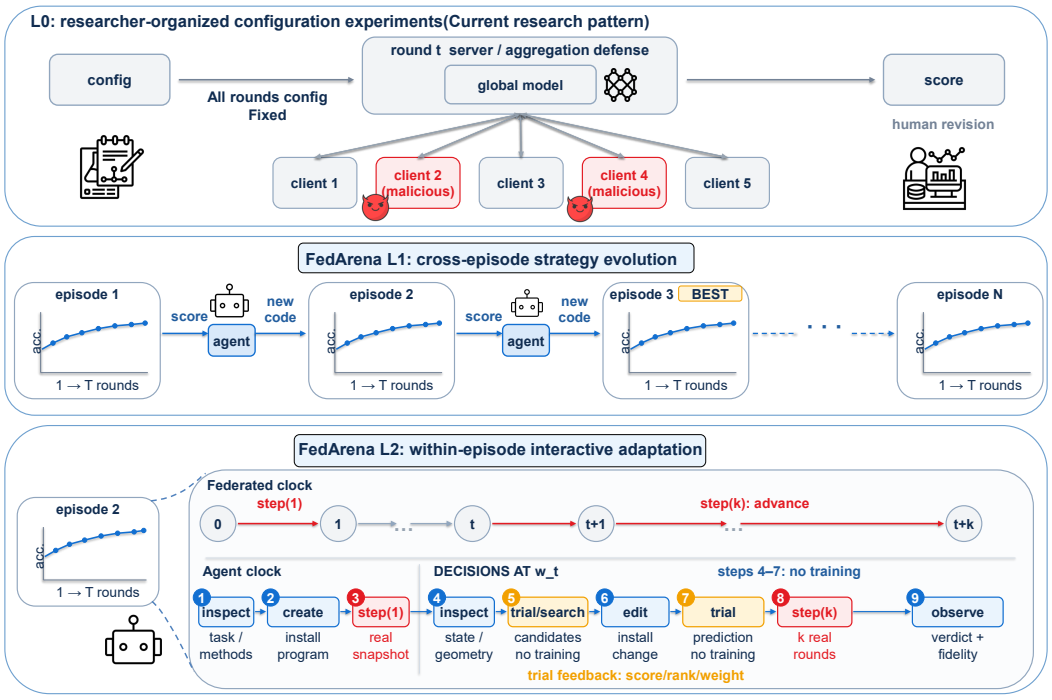

Figure 1: Three research modes: researcher-configured experiments (L0), agent revision between complete evaluations (L1), and intervention and timing decisions within an episode (L2).Labels index interaction granularity, not capability.

environments (Debenedetti et al., 2024; Wang et al., 2026; Carlini et al., 2025). Adaptive evaluation motivates defense-specific testing (Tramèr et al., 2020). Our L2 exposes joint messages, authorized views, local trials, and irreversible rounds, making interventions and observation timing agent decisions.

3 Research Tasks and Search Space

We specify the task, editable objects, and revision boundaries used by the harness.

3.1 Federated Learning and Poisoning

Clients $\mathcal{N} = \{1, \ldots, N\}$ hold datasets D_i and minimize $\mathcal{L}(w) = \sum_i \alpha_i \mathcal{L}_i(w)$, where $\mathcal{L}_i$ is the mean local loss and $\alpha_i = |D_i| / \sum_j |D_j|$ (McMahan et al., 2017). Round-t participants $\mathcal{S}_t$ upload model deltas for server rule A:

$$u_{i,t} = \mathrm{Local}_{i,\xi}(w_t, D_i) - w_t, \qquad w_{t+1} = w_t + \eta_t^{\mathrm{srv}} A(z_t, U_t), \tag{1}$$

Here $U_t = (u_{i,t})_{i \in \mathcal{S}_t}$ and z_t contains model, client, optimizer, and defense state. A T-round run is an *episode*. Strategy p replaces permitted clients' messages, yielding $z_{t+1} = F_\xi(z_t, U_t^p; A, \omega_t)$ under scenario ξ and randomness ω_t.

3.2 Research Tasks, Permissions, and Feedback

A task $\mathcal{Q} = (\xi, \tau, B, J)$ fixes the scenario, threat model, budgets, and outcome evaluation. The contract $\tau = (\mathcal{C}, \mathcal{O}_\tau, \mathcal{V}_\tau, \mathcal{K}_\tau)$ specifies controlled clients, observations, editable surfaces, and sharing. Evaluation $e = \mathrm{Eval}_{\xi,J}(p, A; \Omega)$ records outcomes on seeds Ω; $\mathcal{R}_\tau(e)$ specifies designer feedback separately from strategy inputs. Budgets count complete evaluations, tool calls, local trials, and real rounds.

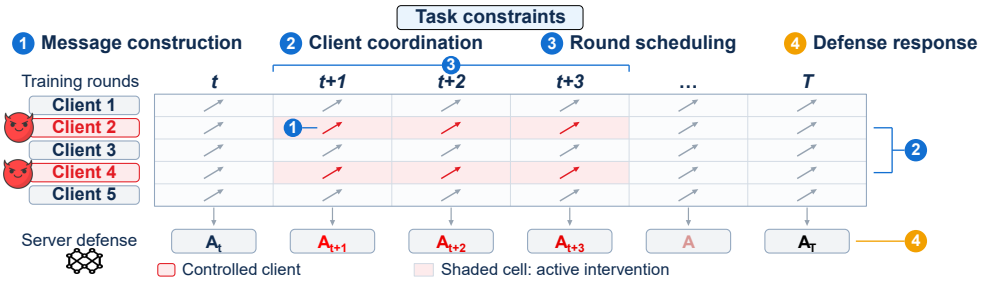

Figure 2: Federated poisoning search space: clients × rounds, with four coupled research dimensions. Red labels mark controlled clients; shaded cells mark active interventions. Each server rule processes its full message column. Task permissions determine the observable and editable parts.

3.3 Search dimensions from existing attacks and defenses

An attack $p = (M, \Gamma, \sigma)$ combines a message constructor, a coalition rule, and an intervention schedule (Shejwalkar & Houmansadr, 2021; Xie et al., 2020a; Zhang et al., 2022). A defense A composes evidence extraction, weighting/selection, sanitization, and reduction (Figure 2). Over the corresponding families $\mathcal{M}, \mathcal{G}, \mathcal{H}, \mathcal{D}$, the admissible space is

$$\mathcal{X}_{\xi,\tau} = \{(M, \Gamma, \sigma, A) \in \mathcal{M} \times \mathcal{G} \times \mathcal{H} \times \mathcal{D} : \mathrm{Valid}_{\xi,\tau}(M, \Gamma, \sigma, A) = 1\}, \tag{2}$$

where Valid requires compatible types, permitted observations and controls, and declared sharing. Fixing $A = A_0$ selects attack research; editing the real server rule requires separate authority. Appendix G expands the message and defense equations and maps existing recipes to these objects.

3.4 Research granularities: L0–L2

Table 1 distinguishes who revises the strategy and when feedback becomes available. An L1 program can itself be stateful; L2 additionally places agent decisions within the evolving episode. Appendix H gives the decision processes.

Table 1: Research modes over the same federated design space.

	L0: configuration	L1: strategy evolution	L2: interactive adaptation
Search object	Researcher-selected strategy and settings	Agent-revised logic and structure	Agent-revised interventions and timing
Feedback	Researcher-interpreted results	Complete-evaluation outcomes and diagnostics	Authorized observations, trials, and real outcomes
Boundary	Configured experiment	Complete episodes per candidate	Tool action; zero or multiple rounds
Artifact	Configuration and comparison	Reusable program and search evidence	Program versions and interaction trace

4 FedArena: A Unified Research Harness

FedArena exposes message construction, coalition emission, and timing as editable research choices (Figure 3). L1 evaluates source programs; L2 compiles component edits into permission-checked protocol actions.

4.1 From existing components to research objects

Attack objects separate message construction, coalition emission, and scheduling. Construction combines references and direction terms, then applies ordered constraints; emission assigns messages to clients. Defense objects separate evidence, selection/weighting, sanitization, and reduction. Local trials and real execution share these message constructors and defense executors.

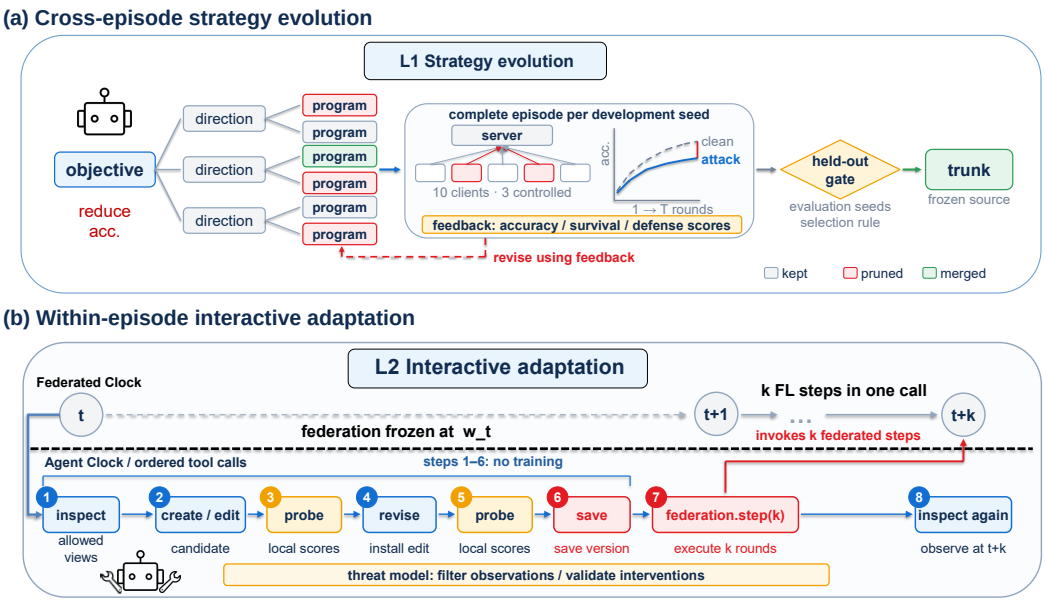

Figure 3: Research interfaces and execution boundaries. (a) L1 evaluates a source version before the next agent revision and retains parent and selection evidence. (b) L2 separates observations, hypothetical trials, installed edits, and real advancement. The threat model controls observations and edits; only advancement executes new rounds. Paths and curves are schematic.

Replacing replicas of a fixed reverse-mean message with norm-preserving, client-specific offsets isolates coalition geometry (Section 5.2). The catalog compiles 14 attack recipes, 10 defense recipes, and 42 primitives into these objects; Appendices G and N give mappings and reference conventions.

4.2 Permissions as Experimental Controls

Inspection filters client identities, messages, statistics, and defensive feedback. Validation checks referenced clients, editable surfaces, and sharing before installation: coalition control does not grant access to other clients' updates. Hypothetical defenses affect only trials unless server edits are separately authorized. Information views therefore vary access within a fixed training scenario.

4.3 Cross-Episode Strategy Evolution

An L1 candidate stores source, a hypothesis, a parent, and evaluator inputs. Complete evaluations return outcomes and defensive diagnostics. In our case, rule-based suggestions guide an agent-implemented revision with an ablation switch. Records retain source versions, seeds, diagnostics, ablations, and selection outcomes, supporting frozen-program execution (Appendix H).

4.4 Within-Episode Interactive Adaptation

Local trials and real execution. A bootstrap round supplies a snapshot. Probe evaluates a temporary candidate on disclosed messages, statistical samples, or a coalition-local proxy, recording the population, defense parameters, snapshot, and candidate. Search compares structures or scalar values, charging each evaluation to the trial budget. Both preserve the installed intervention and training state.

Editing installs a validated version; advancement obtains fresh updates and executes it against the real defense. Identical trial inputs give identical results, but changing the population or snapshot can change the verdict (Section 5.5).

Two clocks. Let m count agent actions and t_m completed federated rounds. Then

$$t_{m+1} = \begin{cases} t_m, & a_m \in \{\text{observe, revise, probe/search, save}\}, \\ t_m + k_m^{\text{exec}}, & a_m = \text{advance}(k_m), \end{cases} \tag{3}$$

where k_m^{exec} counts completed rounds. The task prescribes intervals or lets the agent choose within budget. Later edits do not undo completed rounds; local comparisons and real advancement consume distinct resources.

4.5 Tracing strategy revisions and outcomes

Trials identify candidates and authorized populations; executed rounds identify installed versions. Joining them distinguishes unexecuted proposals, incorrect predictions, and admitted interventions with little damage, measured against same-seed clean runs. The verifier checks permissions, budgets, and event lifecycles; CPU tests and record-replay ablations evaluate these checks separately from attack effectiveness (Appendices K and O).

5 Empirical Studies with FedArena

We examine component choices, diagnostic-guided revision, and the relation between local feedback and training damage.

5.1 Experimental setup

The L1 case uses CIFAR-10/CNN/DnC with three malicious clients among ten; L2 uses MNIST/LeNet-5/Krum with two among ten. Both run for 20 rounds. Three feedback workflows and four information views each have five GPT-5.5 episodes (35 total), alongside a five-seed fixed-program reference. Feedback comparisons pair seeds. The oracle-informed L1 case supplies updates and gradients from client evaluation splits; frozen replay reuses its five validation seeds (Appendix A). Appendix D reports the component matrix.

Damage is same-seed clean minus attacked accuracy, in percentage points (pp), at the endpoint and averaged over rounds. We separately report admission, input-plus-output tokens, and candidate-episode time excluding the clean run. Summaries show mean $\pm$ sample SD; points denote seeds or episodes. Appendices E and M give complete measurements.

5.2 Compositional changes with defense-specific effects

Question: do the editable components isolate consequential research choices? In the 105-trajectory matrix, a coalition comparison changes only emission mode and offset amplitude; direction, constraints, and schedule match. Norm-preserving perturbed emission raises FoolsGold's mean coalition weight from zero to 0.425 and accuracy drop from 0.07 to 84.43 pp. Krum rejects both constructions; trimmed-mean drops remain below 1 pp (Appendices D and Q).

Across 25 search-enabled episodes, agents perform 219 candidate evaluations. Figure 4 follows five structural comparisons and seven norm settings on one snapshot. After installation of a locally accepted construction, controlled messages are selected in 17/20 real rounds.

Finding. One coalition edit has defense-specific effects; agents combine structural choices and numerical refinement in the same interface.

5.3 From evaluator guidance to a tested program revision

Question: how does diagnostic feedback become an executable, testable revision? Of 139 proposed candidates, 121 receive development evaluations (Figure 5a). We follow a *parent strategy* and its *memory-augmented revision* (lineage and selection in Appendix B).

The parent combines a defense-aware direction with a loss-increasing residual. Rule-based diagnostics suggest accumulating a sign-consistent bias; the agent implements sign alignment, exponential

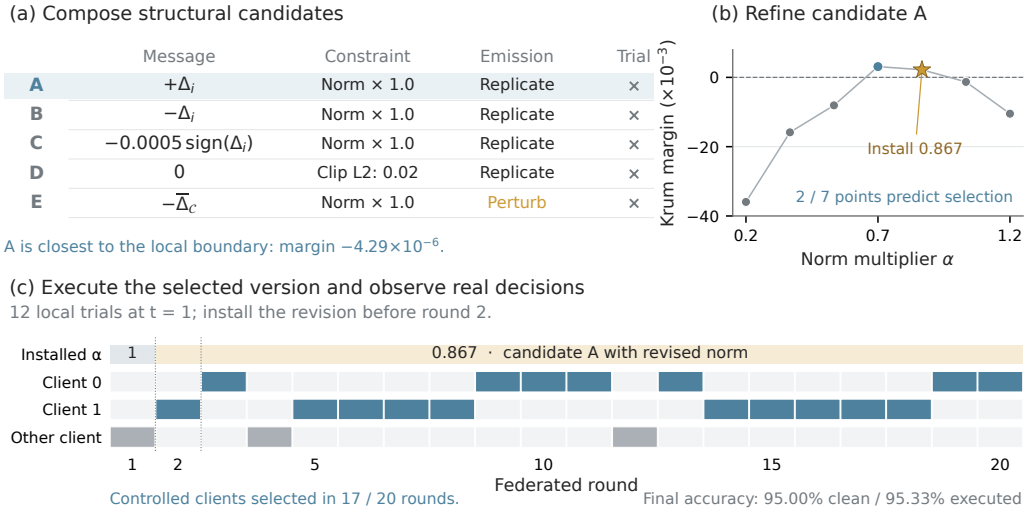

Figure 4: Construction, refinement, and execution. (a) Expressions before constraints: Δ_i is the current update, $\overline{\Delta}_C$ the coalition mean, and $\times$ predicted rejection. (b) Measured norm multipliers relative to own-message norms, tested on a round-1 statistical population; the star marks installation. (c) Actual selections and paired endpoints under the prescribed 1/1/18 cadence.

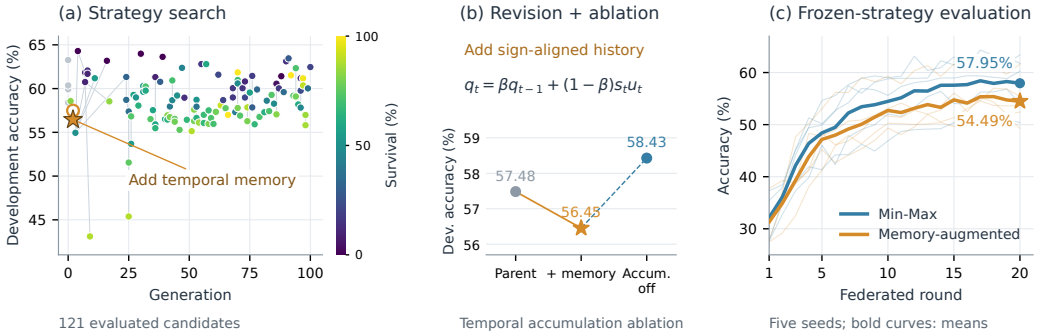

Figure 5: Strategy search, revision, and reuse. (a) Candidate evaluations and parent links, colored by controlled-update survival; gray denotes missing diagnostics. (b) Parent, memory-augmented strategy, and accumulation-disabled ablation. (c) Frozen-strategy evaluation on five paired seeds: thin lines show runs, bold lines means. Gold stars mark the memory-augmented strategy throughout.

accumulation, and an ablation switch. Development accuracy changes from 57.48% to 56.45%; disabling accumulation gives 58.43% (Figure 5b; mechanism in Appendix C).

Frozen replay yields $54.49 \pm 4.57\%$ accuracy versus Min-Max's $57.95 \pm 5.53\%$, lower on four of five seeds (Figure 5c). Appendix B reports paired uncertainty under this oracle-informed setting.

Finding. Diagnostic guidance becomes a tested temporal mechanism: disabling accumulation raises development accuracy by 1.98 pp. Frozen replay separately compares the resulting program with Min-Max.

5.4 Feedback-driven joint intervention

Question: which constructions accompany high admission in the feedback workflows? We compare fixed-program, one-shot, round-feedback, and probe-plus-round workflows with their specified tools, prompts, and revision opportunities (Appendix A; all twenty episodes in Figure 6a). Agent workflows achieve 36%, 67%, and 89% mean admission, with final drops of 0.32, 1.34,

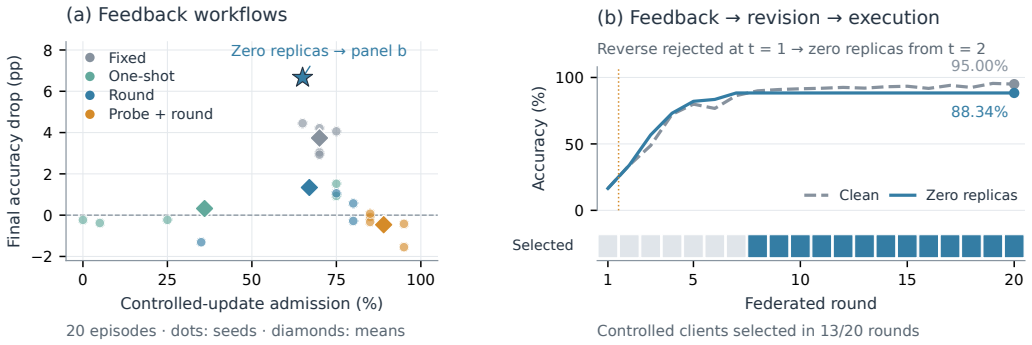

Figure 6: Feedback workflows and intervention. (a) Admission versus accuracy drop: dots are episodes, diamonds means. The star identifies the zero-message revision shown in (b), where zero replicas replace reverse updates from round 2. Curves show paired trajectories; blue cells mark controlled-client selections. The prescribed cadence is 1/1/18.

and -0.46 pp. The fixed program yields 3.74 pp; full-population Min-Max yields 7.89 pp in the component matrix (Appendix D).

All five probe-plus-round episodes finish with nonnegative current/coalition terms and norm matching; the fixed program uploads 0.1 times the coalition mean without norm matching. In Figure 4, the agent retains a current-direction construction closest to the local boundary, then refines its magnitude. Four round-feedback episodes also finish with nonnegative terms; the fifth uses a zero expression (Appendix P).

In this *zero-message revision*, rejected reverse-direction messages are replaced with zero replicas, selected in 13/20 rounds. Final accuracy is 95.00% clean versus 88.34% attacked (Figure 6b).

Mean costs are 40.6k/96.2k/158.2k tokens and 99.7/130.9/145.9 seconds across the three agent workflows (Appendix M).

Finding. High admission accompanies update-preserving choices, not stronger poisoning: all three agent workflows yield less mean damage than the fixed program.

5.5 Information-conditioned exploration and observation timing

Question: how does authorized information change observation timing, cost, and attack outcomes? We vary own-message, population-summary, population-statistics, and full-message views under the same tools and task. Trials use coalition proxies, statistical populations, or disclosed messages as authorized. Here the agent chooses advancement intervals, with up to six decisions for 20 rounds.

Figure 7 shows all twenty schedules; Table 2 reports outcomes. Mean advancement counts are 5.2, 5.2, 3.8, and 3.6; token use is 355k, 334k, 195k, and 204k. The narrower views entail more interactions in these runs, while damage does not improve monotonically.

Matched versions expose a feedback failure: all 28 coalition-proxy predictions anticipate selection, but only six transfer to the next round. The local population has two original updates and two candidate messages, using Krum with $b = 0$ rather than the server's $b = 2$. Statistical and disclosed-message predictions agree in 9/11 and 4/5 comparisons (Appendix P).

A structural explanation. For a proxy (h_1, h_2, v, v) with $b = 0$, each replica's Krum score is $\min_i \|v - h_i\|^2$; neither honest proxy scores lower. For distinct h_1, h_2, v, a replica is selected at any magnitude. Full ten-client Krum instead includes five honest neighbors in each replica's score. On 25 trained snapshots, all 425 proxy predictions are positive, but full-population Krum rejects 334 candidates (proof and same-snapshot controls in Appendix R).

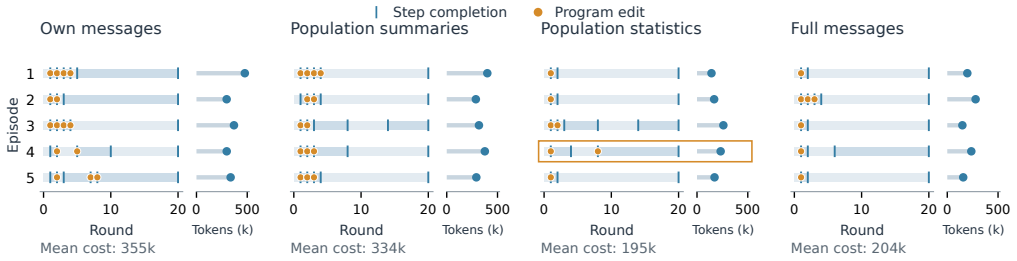

Figure 7: Observation timing and token cost by information view. Rows index episodes within each view; blue ticks mark step completion, gold dots edits, and alternating bands execution batches. The highlighted statistics-view episode uses a 1/3/4/12 cadence, editing after rounds 1 and 8.

Finding. The information view changes the local decision problem, not merely how much feedback is returned. Population and rule assumptions explain a systematic failure that timing and admission summaries alone do not reveal.

Table 2: Information-view outcomes ($n = 5$, mean $\pm$ SD). Admission: %; accuracy drops: pp.

View	Admission	Final drop	AUC drop
Own messages	56.0 ± 37.7	0.05 ± 1.13	0.47 ± 1.22
Population summaries	30.0 ± 35.2	-0.03 ± 0.72	0.54 ± 0.66
Population statistics	91.0 ± 5.5	-0.47 ± 0.74	0.24 ± 1.64
Full messages	80.0 ± 30.8	-1.06 ± 0.87	-2.21 ± 0.98

6 Discussion and Outlook

Research questions determine feedback granularity. L1 supports diagnostic-guided changes to temporal structure; L2 supports revisions of current joint messages and observation timing. Both can yield dynamic strategies, with agent decisions entering between complete experiments or within the federation.

What the domain harness contributes. Compositional objects isolate coalition geometry; trial populations expose changes to the defender's decision problem. Versioned execution connects these choices to outcomes. The component comparison and proxy analysis demonstrate their research use beyond measuring attack strength alone.

Scope and limitations. The studies characterize research operations rather than rank agent frameworks. The archived L1 case uses evaluator guidance and oracle access; broader scenarios, matched-budget productivity comparisons, and fixed-cadence controls remain further work. Appendices A and D specify selection, data access, and numerical handling.

7 Conclusion

FedArena makes message construction, client coordination, and revision timing executable research choices. Component comparisons reveal defense-specific effects; matched traces and a proxy-score analysis explain why deterministic local feedback can mislead. Together with cross-episode program revision, these studies demonstrate how the harness supports controlled investigation of adaptive federated poisoning.

Reproducibility statement

Appendix A specifies configurations, authorized information, budgets, and metrics. Appendix B reports lineage, the development ablation, and frozen-program results; Appendix C defines the strategy. Appendices F, O, and P specify action records, controlled mutations, and trial–execution matching. Appendices Q and R give the paired component reanalysis, proxy-score proof, and same-snapshot controls. Numerical artifacts retain source hashes and all prescribed candidate comparisons.

Ethics statement

The experiments study robustness in local simulations using public image datasets. They do not interact with production FL deployments. Threat-model restrictions and action records support authorized, auditable research. Automated poisoning research is dual use; the intended application is understanding attack–defense behavior and improving evaluation and defenses.

AI use statement

Generative AI assisted conceptual-framework and formalization discussions, derivation and checking of mathematical arguments, hypothesis refinement, experimental-design feedback, method implementation, result interpretation, literature retrieval, manuscript drafting and editing, and figure preparation. LLM agents also executed the research workflows analyzed in this paper; their recorded observations and actions are experimental data. Verification included source-based citation checks, executable interface tests, and recalculation of reported measurements from archived evaluations and traces. The authors are responsible for the final text, claims, code, and reported results.

References

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, 2020. URL https://proceedings.mlr.press/v108/bagdasaryan20a.html.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/ec1c59141046cd1866bbbcdfb6ae31d4-Abstract.html.

Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020. URL https://arxiv.org/abs/2007.14390.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

Nicholas Carlini, Edoardo Debenedetti, Javier Rando, Milad Nasr, and Florian Tramèr. AutoAdvExBench: Benchmarking autonomous exploitation of adversarial example defenses. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/carlini25a.html.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97091a5177d8dc64b1da8bf3e1f6fb54-Abstract-Datasets_and_Benchmarks_Track.html.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium*, pp. 1605–1622, 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses*, pp. 301–316, 2020. URL https://www.usenix.org/conference/raid2020/presentation/fung.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. doi: 10.1109/ACCESS.2019.2909068.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pp. 5311–5319, 2021. URL https://proceedings.mlr.press/v139/karimireddy21a.html.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning*, pp. 11814–11827, 2022. URL https://proceedings.mlr.press/v162/lai22a.html.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, 2026.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://leon.bottou.org/papers/lecun-98h.

Haoyuan Li, Mathias Funk, and Aaqib Saeed. Helmsman: Autonomous synthesis of federated learning systems via collaborative LLM agents. In *International Conference on Learning Representations*, 2026.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, et al. GEM: A gym for agentic LLMs. In *International Conference on Learning Representations*, 2026.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Milad Nasr, Nicholas Carlini, Chawin Sitawarin, Sander V. Schulhoff, Jamie Hayes, Michael Ilie, Juliette Pluto, Shuang Song, Harsh Chaudhari, Ilia Shumailov, Abhradeep Guha Thakurta, Kai Yuanqing Xiao, Andreas Terzis, and Florian Tramèr. The attacker moves second: Stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections. In *35th USENIX Security Symposium*, 2026. URL https://www.usenix.org/conference/usenixsecurity26/presentation/nasr.

Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022. doi: 10.1109/TSP.2022.3153135. URL https://arxiv.org/abs/1912.13445.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security Symposium*, 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019. URL https://arxiv.org/abs/1911.07963.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, et al. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/328b81881da145412f2bc56c998dfb6a-Abstract-Conference.html.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/11f38f8ecd71867b42433548d1078e38-Abstract.html.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In *International Conference on Learning Representations*, 2026.

Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision*, pp. 3–19, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Yuxin_Wu_Group_Normalization_ECCV_2018_paper.html.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. URL https://arxiv.org/abs/1708.07747.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=rkgyS0VFvr.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 261–270, 2020b. URL https://proceedings.mlr.press/v115/xie20a.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 2018.

Heyi Zhang, Yule Liu, Xinlei He, Jun Wu, Tianshuo Cong, and Xinyi Huang. SoK: Benchmarking poisoning attacks and defenses in federated learning. *arXiv preprint arXiv:2502.03801*, 2025. URL https://arxiv.org/abs/2502.03801.

Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael W. Mahoney, Joseph E. Gonzalez, Kannan Ramchandran, and Prateek Mittal. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*, pp. 26429–26446, 2022. URL https://proceedings.mlr.press/v162/zhang22w.html.

A Experimental Settings and Evaluation Semantics

The exact task instructions and their configured budgets are provided in Appendix I; Appendix H details the corresponding execution semantics.

Federated worlds. Table 3 summarizes the two evaluated training configurations. All clients participate in each round. The MNIST world resizes inputs to 32×32, uses IID partitions, and trains a 61,706-parameter LeNet-5 (LeCun et al., 1998). Its explicit local-step limit is 20 steps per client per round; the configuration also sets one local epoch. The CIFAR-10 world (Krizhevsky, 2009) uses a Dirichlet partition with $\alpha = 0.5$, a CNN, and five local epochs. A round denotes a server–client training exchange, not a local epoch.

Table 3: Training configurations used in the empirical studies.

Setting	L1 search and frozen execution	Component and L2 studies
Dataset / model	CIFAR-10 / CNN	MNIST / LeNet-5
Partition	Non-IID, Dirichlet $\alpha = 0.5$	IID
Clients / controlled	10 / 3	10 / 2
Participation / rounds	10 per round / 20	10 per round / 20
Local training	5 epochs	20-step limit; 1 epoch configured
Learning rate	0.01	0.01
Defense	DnC	Krum; matrix also trimmed mean and FoolsGold
Seeds	Development 0,1; subsequent 2–6	0–4 in each condition

L1 access and selection. The L1 search authorizes full-update observation, local defense probes, and objective-gradient queries. Candidate #680 combines benign-message geometry with the supplied loss-gradient direction. The oracle uses a separate model copy and a fixed batch of up to 256 examples sampled across client test splits. These splits also supply the reported evaluation accuracy: the measurements characterize oracle-informed attack development and execution on those splits. Development uses seeds 0–1; seeds 2–6 inform candidate validation and selection and are reused for the frozen replay. The archive selects #680 as its best candidate after this process; we examine it because its parent, diagnostic guidance, implementation, and accumulation ablation are all recorded. The historical archive does not retain the generating LLM's model identifier. Appendix J identifies the diagnostic source.

L2 agents and budgets. The feedback and information-view experiments use `gpt-5.5`, with no explicit temperature override. Each episode is a separate agent execution. The probe-enabled and information-view tasks allow up to 28 agent turns, 40 tool calls, 12 probes, six advancement decisions, 600,000 total tokens, and 2,400 seconds. Completion requires all 20 rounds and a valid trace; automatic completion of remaining rounds is disabled. The round-feedback task prescribes an initial one-round execution, one revision opportunity, another round, and batched completion. The information-view task asks the agent to choose its own intervals. Thus, the feedback study compares specified workflows, while the information study varies authorized views under a common task.

Fixed-program reference. The scripted reference uploads $0.1\overline{u}_{\mathcal{C},t}$, where $\overline{u}_{\mathcal{C},t}$ is the coalition mean update. Its specification combines that reference with a $-0.9\overline{u}_{\mathcal{C},t}$ direction, with no message constraints. Both controlled clients replicate the leader message, with no emission noise, in every round from round 1. This definition is fixed across all five scripted episodes.

Authorized information. All four information regimes retain the controlled-client summaries, global metrics and model summary, selected-client information, the aggregation rule, and disclosed aggregation decisions. Population summaries additionally expose summaries of all updates; population statistics add disclosed population statistics; the full view additionally permits raw updates. Collusion remains enabled for controlled clients 0 and 1, and the allowed control surface is their uploaded model updates. The base tools and objective remain unchanged across the four views.

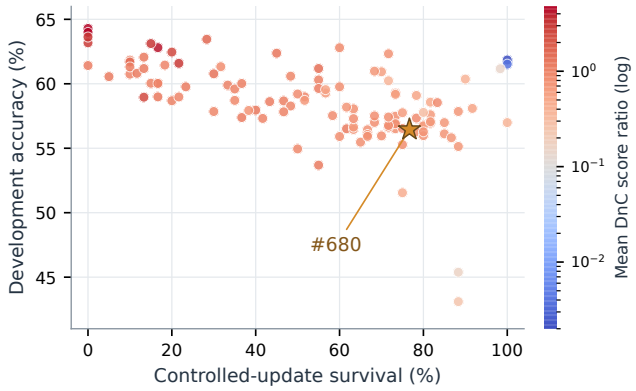

Figure 8: Defensive response and training outcome for 116 run-22 candidates. Color denotes the recorded mean DnC score ratio on a logarithmic scale; the star marks candidate #680.

Pairing and metrics. For a seed s, let $a_{s,t}^{\text{clean}}$ and $a_{s,t}^{\text{attack}}$ denote the logged global accuracy at round t. Final drop is $100(a_{s,T}^{\text{clean}} - a_{s,T}^{\text{attack}})$. The reported AUC metric is the arithmetic mean of logged round accuracies, so its paired drop is $100T^{-1}\sum_{t=1}^{T}(a_{s,t}^{\text{clean}} - a_{s,t}^{\text{attack}})$ for complete T-round traces. Larger drops indicate greater degradation. Admission is the fraction of defended rounds in which modified messages receive influence under the analyzer's defense-specific decision record. Coalition weight measures the aggregate contribution assigned to controlled messages. For Krum's single selection these coincide; for continuous weighting, they describe different quantities. All reported SDs are sample SDs across the five observations, not confidence intervals.

B Search Records and Frozen-Program Evidence

Appendix J connects the representative source revision to its recorded diagnosis, hypothesis, and temporal-state implementation.

The *parent strategy* and *memory-augmented strategy* in Section 5.3 and Figure 5 correspond to candidates #679 and #680, respectively, in run 22. The accumulation-disabled condition sets `accum_strength=0` in the latter program. The archived candidate mean is 54.09%; the subsequent frozen-program evaluation below gives $54.49 \pm 4.57\%$.

The exported run-22 record contains candidate identity, generation, parent, hypothesis, diagnosis, development evaluation, subsequent evaluation where present, ablation record, and source hash. There are 139 entries, 121 development evaluations, and generation values 0–100. Initialization and later revisions appear in the same archive. Figure 5 connects evaluated parents and children rather than interpreting the number of generations as the number of programs.

For 116 candidates, both controlled-update survival and a positive mean DnC score ratio are recorded. Figure 8 places development outcomes alongside these diagnostics, documenting the defensive feedback available during strategy search.

Table 4: Recorded development comparison for the representative modification chain. The temporal-off row is the archived ablation of candidate #680.

Candidate / configuration	Development accuracy (%)
Parent #679	57.4842
Revised #680	56.4543
#680 with `accum_strength=0`	58.4297

Frozen candidate #680 and Min-Max are evaluated under DnC using matched seeds 2–6. Table 5 reports the final accuracies underlying the individual trajectories in Figure 5(c). The frozen source, evaluation settings, and candidate lineage identify the program tested in this comparison.

The paired difference, Min-Max minus candidate accuracy, is 3.46 pp, with a descriptive 95% paired-t interval of $[-1.04, 7.96]$ pp (five pairs, four degrees of freedom). This is the uncertainty of the fixed-program comparison on the reused validation seeds. The development accumulation ablation in Table 4 addresses a different comparison within the revised program; the five-seed table does not isolate that component.

Table 5: Frozen-program execution: final accuracy in percent, lower indicating greater degradation.

Seed	Min-Max	Candidate #680
2	60.2294	53.6996
3	50.3634	52.4059
4	54.0079	49.3094
5	63.3818	61.6227
6	61.7610	55.3969
Mean ± SD	57.95 ± 5.53	54.49 ± 4.57

C Mechanism of the Representative L1 Artifact

Candidate #680 combines a feasible base direction, an orthogonalized harm residual, temporal accumulation, and a final magnitude search. Let B_t contain benign updates available to the program, h_t be the objective-gradient direction, and b_t the output of the defense-aware base solver. With V_t containing the returned orthonormal benign-subspace directions, form

$$\widetilde{r}_t = (I - V_t^\top V_t) h_t, \tag{4}$$

$$r_t = \widetilde{r}_t - \mathrm{proj}_{b_t}(\widetilde{r}_t), \qquad u_t = r_t / \|r_t\|_2. \tag{5}$$

The implementation handles zero norms and numerical exceptions explicitly. It aligns the residual with the previous accumulator before applying exponential averaging:

$$s_t = \begin{cases} -1, & \langle q_{t-1}, u_t \rangle < 0, \\ 1, & \text{otherwise}, \end{cases} \qquad q_t = \beta q_{t-1} + (1-\beta) s_t u_t. \tag{6}$$

The first valid residual initializes q_t. A once-per-round guard and shared coalition output prevent repeated state updates for the same round. Defaults are $\beta = 0.85$, accumulation strength $\lambda = 0.75$, and base weight $\alpha = 0.2$. The residual direction is augmented with the normalized accumulator, scaled by a bounded round-growth factor $g_t = \min(1.5, 1 + 0.03t)$, and normalized to a benign-residual budget ρ_t:

$$d_t = \alpha b_t + (1-\alpha)\rho_t \frac{u_t + \lambda g_t q_t / \|q_t\|_2}{\|u_t + \lambda g_t q_t / \|q_t\|_2\|_2}. \tag{7}$$

A final defense-aware magnitude search selects the uploaded direction's scale, and controlled clients share the resulting update. These expressions summarize the source; the archived program includes fallback branches. The recorded $\lambda = 0$ ablation tests the accumulation component within the existing design. The case exemplifies a hypothesis becoming an executable temporal mechanism and an associated test.

D Component Interfaces and Evaluation

The component study evaluates seven attack configurations against Krum, trimmed mean, and FoolsGold. In the coalition comparison, both programs share a reverse-mean direction and schedule; the emission/noise configuration differs. We call the perturbed-replica configuration *perturbed coalition* to identify this intervention directly.

Table 6: Implementation mapping for the research operations used in the paper.

Tool family	Contract
`federation.inspect`	Returns authorized protocol state, message geometry, program and method records.
`attack.create/edit`	Validates and versions typed message, coalition, and schedule components.
`federation.probe/search`	Evaluates candidates on the available snapshot and authorized population representation; returns local defensive predictions.
`federation.step`	Applies the installed program during each requested real round, records actual completion, and returns disclosed results.
`attack.save`	Saves a program version and associated hypothesis.
`defense.create/edit`	Edits defensive components when the task authorizes server-side design.

Table 7: Full component-matrix results underlying Figure 9(a). Final-accuracy drop in pp, mean ± sample SD over five paired seeds. † includes one run with non-finite loss and finite accuracy; its accuracy is retained in the five-seed summary.

Attack / emission	Krum	Trimmed mean	FoolsGold
Sign flip	-0.79 ± 0.58	0.76 ± 0.18	1.94 ± 0.32
IPM	5.54 ± 2.22	0.64 ± 0.09	0.07 ± 0.18
ALIE	-0.68 ± 0.53	0.04 ± 0.09	0.07 ± 0.18
Min-Max	7.89 ± 1.94	0.81 ± 0.15	0.08 ± 0.17
Min-Sum	7.79 ± 1.45	0.81 ± 0.16	0.07 ± 0.19
Replicated	-0.79 ± 0.58	0.97 ± 0.23	0.07 ± 0.18
Perturbed	-0.79 ± 0.58	0.86 ± 0.17	$84.43 \pm 3.22^{\dagger}$

Controlled component comparison. Figure 9 connects the full matrix to a controlled emission change. Replicated and perturbed coalition programs share their message formula, schedule, membership, and local-training settings; emission mode and perturbation amplitude change. Under FoolsGold, mean coalition weight rises from approximately zero to 0.425 and final accuracy drop from 0.07 to 84.43 pp, while Krum rejects both configurations. Replicated messages closely track clean FoolsGold training; perturbed emission keeps accuracy low. Table 7 retains all means and sample standard deviations.

Numerical failures. One of the five perturbed-coalition/FoolsGold runs produces non-finite loss. Its recorded accuracy remains finite and is included in the five-seed accuracy statistics, including the 84.43-pp mean drop; non-finite loss summaries are treated as missing. Figure 9 and Table 7 mark this condition with †.

E Additional Interaction Results

The summaries below are expanded into every seed's outcomes, research operations, execution schedule, and complete training trajectory in Appendix L.

Table 8: Feedback-workflow results over five paired seeds. Accuracy drops are in pp; admission is in percent. Token counts are workflow means.

Workflow	Final drop	AUC drop	Admission	Tokens (k)
Fixed program	3.74 ± 0.70	2.53 ± 1.20	70 ± 4	–
One-shot	0.32 ± 0.85	0.45 ± 1.02	36 ± 37	40.6
Round feedback	1.34 ± 3.11	1.41 ± 0.65	67 ± 19	96.2
Probe + round	-0.46 ± 0.64	0.76 ± 0.27	89 ± 5	158.2

The four views change the information available to tool operations and trial construction, rather than only changing a verbal description in a prompt. Admission is non-monotone across the views, and

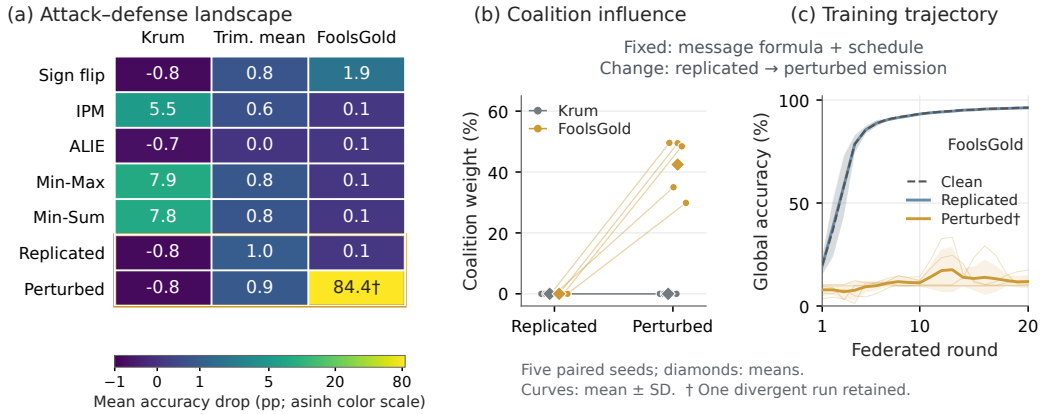

Figure 9: Supporting component study. (a) Mean final-accuracy drops over five seeds per cell; a continuous asinh color scale shows the full numerical range. (b) Same-seed coalition weights under replicated and perturbed emission; diamonds mark means. (c) FoolsGold accuracy trajectories, mean ± SD, with individual perturbed trajectories in faint lines. All observations are retained, including one numerically divergent run (†).

Table 9: Information-view decision and cost measurements (mean ± SD).

View	Advancement decisions	Tokens (k)
Own messages	5.20 ± 0.84	355.43 ± 73.92
Population summaries	5.20 ± 0.45	334.03 ± 50.84
Population statistics	3.80 ± 1.30	195.26 ± 48.84
Full messages	3.60 ± 0.89	203.75 ± 54.77

endpoint degradation remains small. Their behavioral differences therefore characterize information-conditioned exploration and resource use, with efficacy measured separately. The full numerical exports retain every seed and condition.

F Trace Interpretation and Verification

Appendix K complements the recorded-episode checks with executable fixtures for state, authority, versioning, and budget accounting.

Table 10 counts successful inspect, edit, probe/search, and step calls. Create/save and failed calls are excluded, so it is a composition of research operations rather than a count of all model interactions. One search call may evaluate several candidates and consume several probes. Likewise, an agent turn, a tool call, and a real advancement decision are distinct events.

The round-feedback seed-2 example is selected for its interpretable revision and high observed effect. Its new program is installed after round 1 and applies from round 2; this does not mean Krum immediately selects the new message in round 2. The plotted verdict sequence shows subsequent selections.

Recorded local search in Figure 4. The probe-plus-round seed-2 trace is selected to expose the complete structural-search, scalar-refinement, and installation sequence. The five structural candidates and seven scalar samples are agent-proposed, not a researcher-prescribed sweep. All five structural candidates are predicted rejected; the current-direction replica has margin -4.29×10^{-6}, closest to the local boundary. Scalar multipliers 0.700 and 0.867 predict selection, and the installed patch uses 0.8666666667. Search uses the disclosed Krum rule on a statistical population derived from the authorized view. The figure connects only evaluated points, without inferring a densely measured feasible region. The twelve evaluations exhaust the probe budget; a subsequent direct probe is rejected and contributes no plotted point. Real round 2 selects client 1. Actual controlled selections occur

Table 10: Mean successful research-tool calls per episode (five episodes per agent workflow).

Workflow	Observe	Revise	Probe/search	Advance
One-shot	2.4	0.0	0.0	1.0
Round	5.0	1.0	0.0	3.0
Probe + round	4.4	1.2	2.4	3.0

in 17 of 20 rounds, with clean/executed final accuracies 95.00%/95.33%. The task prescribes the $1/1/18$ execution cadence.

Across the two interaction suites, successful search receipts in 25 episodes account for 219 candidate evaluations; each count is matched to a persisted `search_evaluation` record. Population representations comprise 95 statistical, 70 coalition-proxy, and 54 observed-population evaluations. These are local trial counts, not counts of unique algorithms. The exported audit retains episode, seed, snapshot round, searched value, candidate scores/ranks, and predicted outcomes for each trial.

The statistics-view seed-3 example is selected for nonuniform timing and a feedback-linked change from coefficient 0.5 to 0.75. All twenty schedules appear in Figure 7; the exemplar is not used to estimate a population effect on its own. Multiple edits at the same completed-round position share a marker. A receipt is evidence of an executed edit, not automatically evidence of a distinct algorithmic idea.

Verification records whether the configured action, permission, budget, and completion checks pass; all twenty information-view episodes have a valid result. Probe records are associated with the authorized population and candidate version. A local defensive prediction is evaluated on its specified snapshot and can differ from a later real verdict as training changes the messages. These distinctions are explicit in both the logs and the two-clock semantics.

G Search-Space Instantiations

This section instantiates the search dimensions of Section 3.3 with implemented attack and defense compositions. A named recipe compiles into the same typed objects that an agent can edit; its name provides an initialization rather than a separate execution path. Consequently, changing a direction, coalition emission, or selection stage exposes a research decision inside a familiar algorithm.

G.1 Message and defense equations

For $p = (M, \Gamma, \sigma)$, the coordination rule Γ produces client roles or shared directions $q_{i,t}$ from authorized history h_t^τ. A binary schedule $\sigma_{i,t}$ selects interventions:

$$u_{i,t}^p = \begin{cases} M(q_{i,t}, h_t^\tau), & i \in \mathcal{C} \cap \mathcal{S}_t \text{ and } \sigma_{i,t}(h_t^\tau) = 1, \\ u_{i,t}, & \text{otherwise.} \end{cases} \tag{8}$$

Evidence extraction E, selection/weighting W, sanitization K, and reduction R decompose a defense as

$$\begin{aligned} e_t^{\text{def}} &= E(z_t, U_t^p), & \beta_t &= W(e_t^{\text{def}}), \\ \widetilde{U}_t &= K(U_t^p, e_t^{\text{def}}, \beta_t), & A(z_t, U_t^p) &= R(\widetilde{U}_t, \beta_t). \end{aligned} \tag{9}$$

Unused stages are identities. A fixed-defense task permits attack revisions while retaining $A = A_0$; replacing the actual aggregation rule requires server-control authority.

G.2 Message construction and coalition composition

Let μ_t and s_t denote the coordinate-wise mean and standard deviation of an authorized reference population. A message program combines a reference, direction terms, and ordered constraints before coalition emission. Examples include ALIE's $\mu_t + zs_t$, IPM's $-\epsilon\mu_t$, and a sign-flip of the current update. Min-Max and Min-Sum additionally constrain distance to the reference population. Population-dependent recipes require the corresponding update access; local constructions can instead

use controlled-client observations. Coalition emission determines whether controlled clients upload independent, replicated, or perturbed messages, while the schedule determines when the construction is active.

Table 11 lists the fourteen registered attack recipes. C identifies equivalence of the recipe's core operation, while S identifies a structural adaptation using harness components. These labels describe the implementation mapping, independently of the experimental evaluation in Appendix D. Batch- and local-training-based recipes expose additional client-side controls, whereas the reported agent studies use the model-message slice of the space.

Table 11: Attack recipes as composable research objects. C denotes the recipe's core operation; S denotes a structural adaptation.

Recipe	Executable composition	Required authority
alie (C) (Baruch et al., 2019)	reference=mean(benign); direction=z*std(benign); replicated coalition emission	coalition sharing; message control; population updates
badnets (C) (Gu et al., 2019)	select samples; implant trigger; rewrite labels; ordinary local training	batch control
dba (S) (Xie et al., 2020a)	coalition role assignment; partitioned local triggers; independent scaled emission	coalition sharing; batch control; message control
decorrelated_coalition (S)	reference=zero; direction=-strength*mean(coalition); perturbed-leader emission; optional cosine alignment to the coalition mean	coalition sharing; message control
fang (S) (Fang et al., 2020)	reference=zero; direction=-lambda*sign(mean(coalition)); replicated coalition emission; lambda searched with federation.probe, not baked in	coalition sharing; message control
gaussian (C)	reference=gaussian; independent emission	message control
ipm (C) (Xie et al., 2020b)	reference=zero; direction=-mean(benign); replicated coalition emission	coalition sharing; message control; population updates
label_flipping (C)	select samples; rewrite labels; ordinary local training	batch control
mimic (C)	reference=client(id); replicated coalition emission	coalition sharing; message control; population updates
minmax (C) (Shejwalkar & Houmansadr, 2021)	reference=mean(benign); direction=-unit(mean); constraint=max-L2 envelope	coalition sharing; message control; population updates
minsum (C) (Shejwalkar & Houmansadr, 2021)	reference=mean(benign); direction=-unit(mean); constraint=sum-L2-squared envelope	coalition sharing; message control; population updates
model_replacement (S) (Bagdasaryan et al., 2020)	poison batch; task+stealth loss; scale client message	batch control; local-training control; message control
neurotoxin (S) (Zhang et al., 2022)	poison batch; mask local gradient support; project model around broadcast	batch control; local-training control
sign_flipping (C)	reference=zero; direction=-current; independent emission	message control

G.3 Defensive evidence and aggregation

Table 12 maps the ten registered defenses to the stages E, W, K, R. A stage that is unnecessary for a particular rule is an identity. For example, coordinate-wise median is a reduction, clipping modifies messages before reduction, and Krum uses a geometric score followed by selection. Exposing the stages preserves their different response signals: distance scores, retained sets, weights, and sanitized messages provide distinct observations.

Table 12: Defense recipes as composable research objects. C denotes the recipe's core operation; S denotes a structural adaptation.

Recipe	Executable composition	Required authority
coordinate_median (C) (Yin et al., 2018)	reduce=coordinate median	server-rule control
direction_clustering (S)	evidence=cluster membership; decision=keep the largest cluster; sanitize=median L2 clip; reduce=mean	server-rule control
fedavg (C) (McMahan et al., 2017)	reduce=mean	server-rule control
foolsgold (C) (Fung et al., 2020)	evidence=max pairwise cosine; weight=FoolsGold logit reweighting; reduce=weighted mean	server-rule control
history_consistency (S)	evidence=relative shift from the client's last message; decision=keep below a threshold; reduce=mean	server-rule control
krum (C) (Blanchard et al., 2017)	evidence=Krum score; decision=lowest one; reduce=mean	server-rule control
multi_krum (C) (Blanchard et al., 2017)	evidence=Krum score; decision=lowest k; reduce=mean	server-rule control
norm_clipping (C) (Sun et al., 2019)	sanitize=L2 clip; sanitize=optional noise; reduce=mean	server-rule control
rfa (C) (Pillutla et al., 2022)	reduce=geometric median	server-rule control
trimmed_mean (C) (Yin et al., 2018)	reduce=trimmed mean	server-rule control

For Krum, the implementation sums squared Euclidean distances to the $n - f - 2$ nearest other messages and selects the lowest score. This convention follows the algorithmic definition used by the harness. The vendored FLPoison implementation instead uses Euclidean distances and $n - f - 1$ neighbors; implementation parity is therefore recorded separately from the mathematical recipe mapping. The reference checks include identical outputs for coordinate median and trimmed mean on their audited fixtures, and an explicit decision-difference fixture for Krum.

G.4 From a recipe to a research object

Listing 1 compiles and validates one attack and one defense. Compilation normalizes parameters and produces a typed specification; validation then checks it against an explicit threat contract. Server-rule control is granted in this example to illustrate defense-design access. In the fixed-defense agent studies, the server rule is harness-owned instead.

Listing 1: Compilation and validation of attack and defense objects.

```
from fedarena.recipes import build_default_recipe_catalog
from fedarena.primitives import build_default_registry
from fedarena.programs import validate_program, validate_defense_program
from fedarena.specs import ThreatModel

catalog = build_default_recipe_catalog()
registry = build_default_registry()
threat = ThreatModel(
    malicious_client_ids=(0, 1),
    observable_fields=frozenset({"all_updates"}),
    control_surfaces=frozenset({"malicious_updates", "server_aggregation"
    }),
    collusion=True,
)
attack = catalog.compile(
    "ipm", target_client_ids=(0, 1),
    parameters={"scaling_factor": 0.5}, seed=0,
)
defense = catalog.compile("krum", parameters={"assumed_byzantine": 2})
assert validate_program(attack, threat, registry).valid
assert validate_defense_program(defense, threat, registry).valid
```

Extending the catalog requires a parameter schema, declared authority requirements, and a compiler into these objects. The resulting object is checked through the same validation and execution path. A new primitive additionally supplies the operation performed at its declared client or server stage. This separates naming a method from defining the operations that an agent can investigate. Appendix N lists the full primitive vocabulary, configuration inventory, and tool input/return structures.

H Interface Semantics and Research Loops

This section expands the execution semantics in Section 4. Both interfaces connect a candidate to its authorized inputs, execution settings, and observations. Their difference is the point at which a new agent decision can revise the object under investigation.

H.1 Complete-experiment strategy evolution

With history H_j of candidates, settings, and evaluation evidence, agent π_1 proposes $x_j = (p_j, A_j)$:

$$x_j \sim \pi_1(\cdot \mid H_j), \qquad e_j = \mathrm{Eval}_{\xi,J}(p_j, A_j; \Omega_j),$$
$$H_{j+1} = H_j \parallel (x_j, \Omega_j, \mathcal{R}_\tau(e_j)). \tag{10}$$

Here $\parallel$ appends evidence. Evaluation budgets count complete candidate executions; the agent's next revision follows the configured evaluation boundary.

An L1 proposal packages source code with a hypothesis and a component-level ablation. The evaluator executes the candidate using the declared scenario and returns training outcomes and defensive diagnostics. A refinement request includes the same candidate's code and its measured behavior; a new proposal can instead introduce a different mechanism. Parent links distinguish these operations in the evidence history.

Listing 2: L1 research loop (semantic pseudocode).

```
Inputs: scenario, threat contract, seed policy, evaluation budget
History <- initialized candidates and measured baseline results
while evaluation budget remains:
    Context <- select parents, observations, and prior hypotheses
    Candidate <- agent proposes or revises source using Context
    Validate the code interface and authorized control surface
    Evidence <- execute complete episodes on the designated seeds
```

```
    Record source version, parent, settings, outcomes, diagnostics
    If scheduled, execute Candidate with its ablation switch off
    Apply the configured validation and candidate-selection rule
    Append the candidate and all evaluation evidence to History
Output: selected source, its configuration, and linked evidence
```

A candidate may maintain temporal state or coordinate multiple controlled clients during execution. Those behaviors are part of the program being evaluated. The agent's next structural revision follows the complete evaluation, so the search can compare alternative implementations of the cross-round rule.

H.2 Within-episode interactive adaptation

At action m, let $s_m = (t_m, z_{t_m+1}, p_m, v_m, b_m)$ record completed rounds, federation state, installed strategy, trial snapshot/proxy, and remaining budgets. With authorized interaction history h_m^τ,

$$\begin{aligned} a_m &\sim \pi_2(\cdot \mid h_m^\tau), & s_{m+1} &\sim \mathcal{T}_{\mathcal{Q}}(\cdot \mid s_m, a_m), \\ o_{m+1} &= \mathcal{O}_{\tau,a_m}(s_m, s_{m+1}), & h_{m+1}^\tau &= h_m^\tau \,\|\, (a_m, o_{m+1}). \end{aligned} \tag{11}$$

The task accounts for resource use through $\sum_m c(s_m, a_m) \preceq B$. Observations are filtered by the action and threat contract; the real-round clock follows Equation 3.

L2 exposes installed program versions, authorized observations, local trials, and real advancement. The completed-round counter changes only when training advances. A trial may update the evidence ledger and consume query budget while leaving the federation model unchanged. Editing likewise changes the installed program for future uploads, without retrospectively changing earlier rounds.

Table 13: State effects of the L2 actions used in the empirical studies.

Action	Federation state	Program, evidence, and budgets
Inspect	Unchanged	Returns a permission-filtered view; records the call.
Create / edit	Unchanged	Validates and installs a version with a future effective round.
Probe	Unchanged	Records a local defensive prediction; spends one trial budget unit.
Search	Unchanged	Records evaluated alternatives; charges each evaluated candidate as a trial.
Advance k	Executes up to k rounds	Uses installed versions, records actual outcomes and completion.
Save	Unchanged	Preserves the installed method and hypothesis.

Listing 3: L2 research loop (semantic pseudocode).

```
Inspect the task's protocol and authorized client controls
Create a permitted joint-message program and a hypothesis
Advance a real round to obtain a protocol snapshot
while the episode is incomplete and action budget remains:
    Observe disclosed state and current defensive response
    Optionally compare candidate structures or scalar settings
        on the authorized snapshot, without advancing training
    Revise the installed program when the evidence motivates it
    Choose an allowed advancement interval k
    Execute k real rounds and obtain the disclosed new evidence
Save the installed method and its hypothesis
The harness computes paired outcomes and verifies the trace
```

Task instructions can restrict this loop, for example to one initial proposal or a prescribed advancement schedule. The information-view study leaves interval choice to the agent within its advancement budget. Thus, the interface exposes timing as an action while the task defines which timing decisions are available in an experiment.

H.3 Population views and local judgement

Local trials operate on one of three authorized population representations: disclosed messages, a population synthesized from disclosed statistics, or a coalition-local proxy. The result records the representation together with candidate scores and selections. This information identifies what was locally tested. A subsequent real verdict is a separate observation produced by actual client training and server aggregation.

For a hypothetical candidate, the trial uses a temporary program instance. It does not replace the installed program. An explicit edit is therefore the commitment boundary between a comparison and the strategy used by later real rounds. The action receipt identifies the program version and effective round, while the execution receipt identifies the rounds actually completed.

Listing 4: Execution receipt and disclosed feedback from round-feedback seed 2. Only the displayed fields are retained.


```
{
  "receipt": {
    "operation": "federation.round_executed",
    "round_before": 1,
    "round_after": 2,
    "rounds_requested": 1,
    "rounds_executed": 1
  },
  "metrics": {
    "round": 2,
    "global_loss": 2.2563902080535887,
    "global_accuracy": 0.3397
  },
  "server_selection": [
    4
  ]
}
```


This receipt records a completed second round and selection of client 4. It is distinct from the preceding edit that installed zero replicas for the controlled clients: installation and selection are separate events. The full action sequence is given in Appendix J.

I Task and Prompt Templates

These templates specify the research actions and feedback available through the interfaces in Section 4. The L2 listings reproduce the system and task text from the experiment configurations, with line wrapping for readability. The L1 listings document the implemented proposal/refinement interface; the historical case's recorded hypothesis and diagnosis are presented separately in Appendix J.

I.1 L1 proposal and refinement contracts

The proposal interface requests a mechanism, a quantitative prediction, an ablation switch, and executable source. Its dynamic context supplies the target defense, selected parents, their measured outcomes, and accumulated evidence. The refinement interface instead supplies the current candidate and its own diagnostics, directing the agent to revise that mechanism.

Listing 5: Implemented L1 output and strategy-code contract (excerpt).

```
## Output: a single JSON object, nothing else
{
  "hypothesis": "1-2 sentences: the causal idea -- WHY this update
    pattern bypasses the rule's filter",
  "mechanism": "3-6 sentences: precisely how the crafted update achieves
    that",
```

```
  "observable": "the falsifiable prediction: what accuracy you expect and
     why",
  "predicted_acc": 0.00,
  "ablation": {
    "param": "setup() config key that turns your core mechanism OFF",
    "off_value": 0.0,
    "why": "if disabling this barely changes the score, the mechanism is
    NOT the cause"
  },
  "method_name": "arena_attack_<name>",
  "class_name": "ClassName",
  "code": "full strategy.py"
}

## Code contract (platform API names -- use them as-is)
The platform's strategy interface uses the class ``ResearchAttackStrategy
    `` and method ``attack()`` -- these are
fixed API names. Inherit, set ``method_name``, implement ``attack()``:
``attack(self, local_model_params, global_model_params, all_client_params
    =None, round_num=0,
   client_id=0, selected_client_ids=None, malicious_client_ids=None,
    coordination=None,
   defense_probe=None, harm_oracle=None, **kwargs)``
Be robust to empty/None inputs and shape mismatches. Return the SAME dict
     keys/shapes/dtypes; never emit NaN/Inf.
```

Listing 6: Implemented L1 refinement instruction (excerpt).

```
## REFINE TASK -- improve THIS EXACT candidate; do NOT invent a new
mechanism You wrote the strategy below. It was just run against the
REAL target defense and instrumented; the measurements are given.
DIAGNOSE the failure from those numbers and REPAIR THE SAME CODE --
keep its core mechanism/idea and fix the specific measured weakness
(evasion vs. potency vs. accumulation). A brand-new unrelated
mechanism is NOT what this step wants; that is a different step.
Make the smallest change that the measurements say will lower the
target accuracy.
```

Table 14: Dynamic evidence supplied to proposal and refinement requests.

Context	Proposal	Refinement
Program context	Selected parents and their mechanisms	The exact candidate source being revised
Outcome context	Measured reference and candidate scores	Candidate accuracy and reference score
Defense response	Prior evaluated candidates' diagnostics	Candidate survival, score ratios, diagnosis
Research history	Explored approaches and accumulated observations	Evidence motivating a local repair
Requested result	A source-level strategy proposal	A revised version of the same strategy

I.2 L2 FEEDBACK-WORKFLOW TASKS

The one-shot task exposes inspect, create, save, and advance. Round feedback adds editing after observed execution. Probe-plus-round feedback additionally exposes local probe and search. The fixed-program condition uses a scripted control rather than an LLM prompt. Budgets and completion checks accompany the task text in Table 15; all agent tasks require episode completion and a valid trace, with no automatic execution of unrequested remaining rounds.

Table 15: Configured agent budgets. A dash means the corresponding tool is absent from the task.

Task	Turns	Calls	Probes	Advances	Tokens / time
One-shot	12	16	—	1	250k / 20 min
Round feedback	20	28	—	3	400k / 30 min
Probe + round	28	40	12	6	600k / 40 min
Information views	28	40	12	6	600k / 40 min

Listing 7: One-shot condition: system and task instructions.

```
SYSTEM

You are a researcher running an authorized, local federated-learning
robustness benchmark. The public MNIST federation, controlled
clients, and fixed Krum server are simulated by the harness. Operate
only through the typed tools and authority declared by the threat
model.

TASK

This is the one-shot condition. Inspect the protocol and client-
update surface, then create one typed model-poisoning program for
controlled clients 0 and 1 under fixed Krum. You receive no runtime
federation feedback before committing the program. Do not edit it,
probe it, search it, or split the training trajectory into multiple
decisions. Execute all 20 communication rounds in one
federation.step(rounds=20), save the exact installed method, and
finish with a concise summary. Do not claim a clean-vs-attack
effect; paired scoring and verification belong to the harness.
```

Listing 8: Round-feedback condition: system and task instructions.

```
SYSTEM

You are a researcher running an authorized, local federated-learning
robustness benchmark. The public MNIST federation, controlled
clients, and fixed Krum server are simulated by the harness. Operate
only through the typed tools and authority declared by the threat
model.

TASK

Construct a model-poisoning program for the two controlled clients
under the fixed Krum rule. This condition intentionally removes
counterfactual federation.probe and federation.search. You may learn
only by spending real, irreversible federated rounds. Inspect the
protocol and client-update surface, create one typed message
program, and execute exactly one real round. Inspect the disclosed
state and message geometry after that round. If the real server
decision contradicts your hypothesis, make at most one edit. Execute
exactly one more round, then run all remaining rounds in one batched
federation.step. Save the installed method with a falsifiable
hypothesis and finish with an evidence-based summary. Do not claim
the paired effect yourself; the harness owns the same-seed control,
scoring, and verification.
```

Listing 9: Probe-plus-round condition: system and task instructions.

```
SYSTEM

You are a researcher running an authorized, local federated-learning
robustness benchmark. The dataset is public MNIST and every client
and server is simulated in this process. You may operate only on the
two controlled clients declared by the threat model, may observe
only disclosed fields, and cannot modify the harness-owned server
rule. Work through typed federated protocol tools and leave an
auditable hypothesis, action, feedback, and revision trail.

TASK

The server runs a fixed Krum aggregation rule. Test whether a
protocol-native Agent loop can adapt the two controlled clients'
uploaded model-update messages under partial observability and
optimize the declared benchmark objective relative to the same-seed
clean control. The method is not a menu of published names here.
Express it as a typed message program: reference plus direction
terms, constrained support and geometry, coalition
roles/sharing/emission, and a round schedule. Named methods in the
method record are reusable recipes over this common algebra. Start
by inspecting the protocol, available client controls, and method
record. Create one permitted hypothesis. A local judgement needs a
real message population, so execute at least one real round before
using geometry, probe, or search. `federation.probe` and
`federation.search` form the cheap deterministic inner loop: they do
not advance the federation. Use one compact structural search and,
only if it changes the decision, one compact scalar search. Record
one committed probe whose prediction can be scored against the next
server verdict. After that real verdict, stop retuning unless it
directly contradicts the scored probe. `federation.step` is the
irreversible outer loop. The world has 20 rounds but you have at
most 6 decision points. You choose the feedback cadence by passing
`rounds`; spend one round when immediate feedback may change the
next action and batch several when the hypothesis should be allowed
to run. Prefer a short loop: inspect/create, run one real round,
search/probe/edit, run one scored round, then save and execute all
remaining rounds in one batched step. Read the live state if you
need the remaining cadence budget. Complete every round, save the
final installed program with a concise falsifiable hypothesis, then
finish with an evidence-based summary. Do not claim a clean-vs-agent
effect yourself; paired scoring and verification belong to the
harness.
```

I.3 Information-conditioned timing task

The information-view experiment uses the same task and tool set across its four regimes. Permission filtering changes the population information returned to the agent and used in local trials. The task allocates at most six advancement decisions for twenty training rounds and explicitly asks the agent to choose the intervals. This supplies the experimental context for the observation schedules in Figure 7.

Listing 10: Common task for the four authorized information views.

```
SYSTEM

You are a researcher running an authorized, local federated-learning
robustness benchmark. The public MNIST federation and every client
and server are simulated in this process. Operate only on the
controlled clients declared by the threat model, observe only
disclosed fields, and leave an auditable hypothesis, action,
feedback, and revision trail through typed federated protocol tools.

TASK

The server runs fixed Krum. Adapt the controlled clients' uploaded
model-update messages under exactly the knowledge regime disclosed
by federation.inspect. Do not assume access to any message, summary,
statistic, or decision that is not present in that contract. Express
the method as a typed message program: reference and direction
terms, support and constraints, coalition geometry, and schedule.
Start by inspecting the protocol and attack surface, then create one
falsifiable hypothesis. Spend one real round to obtain a message
population. Use the state-preserving probe or search only when its
authorized population can answer a concrete question; record one
committed prediction before the next real verdict. Revise the
installed program only when disclosed evidence contradicts the
hypothesis. The federation has 20 rounds and at most 6 decision
points. Choose the feedback cadence yourself with
federation.step(rounds=...). Batch rounds when another observation
would not change your decision. Complete every round, save the final
program and hypothesis, and finish with an evidence-based summary.
The harness, not you, owns clean pairing, efficacy scoring, and
verification.
```

J Extended Research Traces

The following records expand the examples in Sections 5.2–5.5. Each links an agent-visible observation to a concrete source or message change and its recorded execution. Textual hypotheses are distinguished from measured outcomes; action tables include unsuccessful calls as well as completed ones.

J.1 L1: from defensive diagnosis to temporal state

The parent candidate #679 combines a defense-aware base direction with a loss-increasing residual. Its measured development accuracy is 57.48%, and controlled updates survive DnC at a recorded rate of 73.33%. The diagnosis below is generated by the evaluator's predefined rule `_diagnose_defense`, using survival and available score/alignment summaries. The sign-consistency suggestion is part of that rule, rather than an independently generated agent hypothesis. The child #680 implements a persistent accumulator with sign alignment and exposes an accumulation-strength switch. The case therefore follows diagnostic-guided implementation and testing.

Listing 11: Rule-based evaluator diagnosis returned for candidate #679.

```
colluders SURVIVE DnC (survival 73%); and is aimed at harm -- if
accuracy still does not drop, the per-round magnitude is too small
or it cancels across rounds; accumulate a SIGN-CONSISTENT bias over
rounds or push gamma toward the feasibility limit
```

Listing 12: Recorded hypothesis of candidate #680.

```
The attack was surviving DnC often enough, but its harmful component
```

```
was not coherently accumulating across rounds, so the effect partly
canceled even when the filter kept it. By turning the residual harm
direction into a sign-consistent round-to-round drift while keeping
the same DnC-safe base/residual split, the malicious bias should
compound instead of reset.
```

Listing 13: Temporal-state update in the frozen candidate #680, extracted from its source.

```
if self._last_round != round_num:
    if self.accum_strength > 0 and residual_unit.norm(p=2) > 0:
        signed_residual = residual_unit
        if self._bias_accum is not None and self._bias_accum.shape ==
    residual_unit.shape:
            if torch.dot(self._bias_accum, signed_residual) < 0:
                signed_residual = -signed_residual
            beta = min(0.999, max(0.0, self.accum_ema))
            self._bias_accum = beta * self._bias_accum + (1.0 - beta) *
    signed_residual
        else:
            self._bias_accum = signed_residual.clone()
    self._last_round = round_num

accum_unit = torch.zeros_like(harm_flat)
if self.accum_strength > 0 and self._bias_accum is not None:
    accum_norm = self._bias_accum.norm(p=2)
    if accum_norm > 0:
        accum_unit = self._bias_accum / accum_norm
```

The once-per-round guard updates the memory only once even when the program is invoked for multiple controlled clients. Sign alignment keeps successive residual contributions consistent with the stored accumulator. The later message-construction stage adds the normalized accumulator to the current residual, scaled by accumulation strength and the bounded round-growth factor defined in Appendix C. This turns the hypothesis into an explicit stateful operation rather than a change to a single upload magnitude.

The revised development accuracy is 56.45%, compared with 58.43% when accumulation is disabled. These measurements are the component check in Figure 5(b). Table 5 reports subsequent frozen execution on the same seed set previously used for selection validation. Source lineage, evaluator guidance, the implemented hypothesis, and measured outcomes have separate roles in the research record.

J.2 L2: STRUCTURE SEARCH FOLLOWED BY SCALAR REFINEMENT

The probe-plus-round seed-2 episode starts with a replicated current-update construction. After its first real round, the agent reads geometry and state, evaluates five structural variants, and then evaluates seven scalar variants of the selected construction. Both searches use the same completed-round snapshot and the authorized statistical population.

The five structural variants cover current-direction, reverse-direction, signed-direction, zero-message, and perturbed-coalition constructions. All are locally predicted rejected. The current-direction replica has the closest recorded margin, motivating a norm scan. Multipliers 0.700 and 0.867 are predicted selected; the installed revision uses 0.867. A further probe request is rejected after the twelve-trial budget is exhausted. The agent then executes real round 2, saves the method, and completes the remaining eighteen rounds.

Table 16: Action sequence for the local-search case. t counts completed rounds after the action.

Call	t	Tool	Recorded outcome
1	0	federation.inspect	Authorized protocol view
2	0	federation.inspect	Authorized attack_surface view
3	0	federation.inspect	Authorized methods view
4	0	attack.create	Version effective from round 1
5	1	federation.step	Advance 0 to 1; 1 real rounds
6	1	federation.inspect	Authorized geometry view
7	1	federation.inspect	Authorized state view
8	1	federation.search	5 local candidate evaluations; no round advance
9	1	federation.search	7 local candidate evaluations; no round advance
10	1	attack.edit	Version effective from round 2
11	1	federation.probe	Rejected: ToolBudgetExceeded
12	2	federation.step	Advance 1 to 2; 1 real rounds
13	2	attack.save	Save installed method and hypothesis
14	20	federation.step	Advance 2 to 20; 18 real rounds

The actual second-round verdict selects controlled client 1; across the complete run, controlled messages are selected in 17/20 rounds. Final accuracy is 95.33%, versus 95.00% in the paired clean run. These records substantiate the structural-search, numerical-refinement, and commitment sequence in Figure 4, with training outcome measured separately from local selection predictions.

J.3 L2: REVISING JOINT MESSAGES AFTER REAL FEEDBACK

The zero-message revision highlighted in Figure 6 is the round-feedback episode with recorded seed 2. In round-feedback seed 2, the initial message program emits identical damped reverse updates. The first real verdict selects benign client 4. Following inspection, the agent replaces the message expression with a zero reference and no direction or constraint terms; the existing coalition configuration retains replicated emission. The recorded edit and rationale are shown below.

Listing 14: Message patch and recorded hypothesis for the zero-replica revision.

```
{
  "message": {
    "reference": {
      "source": "zero"
    },
    "directions": [],
    "constraints": []
  },
  "hypothesis": "Round 1 contradicted the damped-reverse selection
    hypothesis: Krum selected benign client 4 while controlled uploads
    were excluded. Edit once to replicated zero updates, betting that
    exact mutual distance plus central small-norm geometry will lower
    Krum score enough for selection; if selected, the aggregate will be a
     null step and slow global improvement under keep_count=1."
}
```

Table 17: Action sequence for the round-feedback case. t counts completed rounds after the action.

Call	t	Tool	Recorded outcome
1	0	federation.inspect	Authorized protocol view
2	0	federation.inspect	Authorized attack_surface view
3	0	federation.inspect	Authorized methods view
4	0	attack.create	Version effective from round 1
5	1	federation.step	Advance 0 to 1; 1 real rounds
6	1	federation.inspect	Authorized state view
7	1	federation.inspect	Authorized geometry view
8	1	attack.edit	Version effective from round 2
9	2	federation.step	Advance 1 to 2; 1 real rounds
10	20	federation.step	Advance 2 to 20; 18 real rounds
11	20	attack.save	Save installed method and hypothesis

The second round still selects benign client 4. Controlled messages are subsequently selected in rounds 8–20, giving thirteen selected rounds in total. The paired accuracy trajectories in Figure 6(b) end at 95.00% clean and 88.34% attacked. The ledger makes the sequence explicit: the agent reacts to observed rejection, installs a new joint construction, and the federation determines whether and when that construction receives influence.

J.4 L2: REVISING THE INTERVENTION AND OBSERVATION INTERVAL

Figure 7 numbers the five episodes within each information view from 1 to 5; these rows correspond to recorded seeds 0–4. The highlighted statistics-view episode is row 4, recorded seed 3. Row numbers align schedules and costs within a view and do not denote paired agent decisions across views. The statistics-view seed-3 episode uses intervals of 1, 3, 4, and 12 rounds. It first installs a current-update construction and then adds a negative coalition-mean term. The coefficient changes from -0.50 after completed round 1 to -0.75 after completed round 8. The program also specifies norm matching and a history-smoothing constraint, so these coefficients describe the direction term before the remaining message transformations.

Listing 15: Recorded rationales for the two message revisions in statistics-view seed 3.

```
AFTER ROUND 1
Round-1 evidence: replicated current was selected by Krum, but pure
negative coalition directions were rejected. Probe predicted current
plus a modest subtraction of the coalition mean remains selected
with larger margin (0.0040 vs 0.0017), so commit that geometry.

AFTER ROUND 8
Round-4 and round-8 verdicts show the committed current-minus-0.5
coalition message is consistently selected by Krum, but global
accuracy improved rapidly, so the efficacy component of the
hypothesis (meaningful degradation while selected) is weak. The last
authorized search predicted current-minus-0.75 remains selected with
positive margin, while >=1.0 is rejected. Commit the strongest
probed selected displacement.
```

Table 18: Action sequence for the information-cadence case. t counts completed rounds after the action.

Call	t	Tool	Recorded outcome
1	0	federation.inspect	Authorized protocol view
2	0	federation.inspect	Authorized attack_surface view
3	0	federation.inspect	Authorized methods view
4	0	attack.create	Version effective from round 1
5	1	federation.step	Advance 0 to 1; 1 real rounds
6	1	federation.inspect	Authorized state view
7	1	federation.inspect	Authorized geometry view
8	1	federation.search	5 local candidate evaluations; no round advance
9	1	federation.search	Rejected: ToolInputError
10	1	attack.edit	Version effective from round 2
11	1	federation.probe	Local prediction recorded; no round advance
12	4	federation.step	Advance 1 to 4; 3 real rounds
13	4	federation.search	6 local candidate evaluations; no round advance
14	8	federation.step	Advance 4 to 8; 4 real rounds
15	8	attack.edit	Version effective from round 9
16	20	federation.step	Advance 8 to 20; 12 real rounds
17	20	attack.save	Save installed method and hypothesis

The two successful searches evaluate five structures at completed round 1 and six structures at completed round 4; a successful probe records a local prediction after the first edit. A rejected search call is retained in the ledger. The next revision occurs after round 8, using the accumulated observations; it is not a new search at that round. The final twelve-round batch completes the episode at 95.07% accuracy, compared with 95.22% clean. This example connects disclosed response, a versioned message change, and the decision to defer the next observation until completion.

K Contract Verification

This section supplies executable checks for the permissions, two-clock semantics, and evidence handling in Section 4.4. The checks use small CPU fixtures and controlled message populations. They test the interface implementation independently of the accuracy measurements in the research episodes.

Table 19: CPU contract checks run for the reported interface implementation.

Test group	Passed	Failed
Local trials and search	68	0
Trace verification	4	0
Tool schemas and budgets	5	0
Registry and reference conventions	4	0
Total	81	0

K.1 State preservation and explicit commitment

The probe tests compare the completed-round counter and global-model digest before and after local evaluation. Hypothetical patches are evaluated without installation, and their local predictions retain the authorized population description. Search is checked to charge each evaluated candidate against the trial budget while preserving the training state. The tests also cover structural and scalar search, non-monotone local responses, and exhausted or insufficient budgets.

K.2 Authorization and observation filtering

Permission fixtures vary coalition membership, sharing, and population visibility. The returned client set is checked against the declared view. Additional tests verify that real-step results do not disclose

benign client records unless the relevant field is explicitly authorized. A trace claiming a broader population than permitted is rejected by the verifier. These checks connect the abstract observation map to the actual information returned through tools.

K.3 Versions, attribution, and execution accounting

Version tests distinguish editing a program from running it again, require a hypothesis when saving a method, and associate scores with the program that actually executed. Batch-advancement checks verify both the requested interval and the number of rounds completed, including termination at the episode horizon. Invalid schemas and unauthorized arguments produce structured errors rather than silently changing the requested experiment.

K.4 Trace integrity and component registration

The trace checks verify complete event lifecycles and detect unauthorized audit records or missing lifecycle events. Replay fixtures compare semantic trace digests under the same restored state; this is an implementation check of recorded execution. Separately, registration checks compile the catalog entries, validate their compatibility declarations, and distinguish reference-implementation parity from the adopted mathematical conventions.

The 81 checks in Table 19 passed in the CPU verification run. In addition, the twenty information-view research episodes passed their configured runtime verifier. The two forms of evidence serve different purposes: controlled fixtures isolate interface properties, while episode verification checks whether the corresponding research execution satisfies its configured contract.

L Complete Interaction Results and Trajectories

This section expands Sections 5.4–5.5 into all forty evaluated episodes: twenty feedback-workflow runs and twenty information-view runs. Every row is linked to its same-seed clean control, and every curve includes all twenty scored rounds. Final drops, trajectory-averaged drops, and admission are recalculated from the saved traces and checked against the reported episode summaries. The tables retain all five seeds in every condition.

L.1 Feedback workflows: every seed and operation sequence

Table 20 reports the paired endpoints, trajectory-averaged effect, defensive admission, advancement count, and execution cost. Positive drops mean lower accuracy under intervention. Token counts are the recorded input-plus-output usage; a dash denotes the scripted control, which does not invoke an LLM. Time is the elapsed episode-run time recorded by the evaluation runner, rather than the paired clean-plus-attack total.

Table 20: Feedback workflows: every evaluated seed. Accuracy and admission are percentages; drops are percentage points.

Seed	Clean	Attacked	Final drop	AUC drop	Admission	Decisions	Tokens (k)	Time (s)
Fixed program								
0	94.83	91.79	3.04	1.17	70	1	–	64.7
1	93.75	90.82	2.93	2.79	70	1	–	64.7
2	95.00	90.79	4.21	1.45	70	1	–	64.7
3	95.22	90.77	4.45	3.23	65	1	–	64.3
4	94.63	90.57	4.06	4.00	75	1	–	64.7
One-shot								
0	94.83	93.90	0.93	-0.06	75	1	39.6	111.9
1	93.75	93.98	-0.23	1.44	25	1	45.9	110.0
2	95.00	95.38	-0.38	-1.04	5	1	39.1	89.3
3	95.22	95.45	-0.23	1.27	0	1	39.2	91.3
4	94.63	93.11	1.52	0.61	75	1	39.1	96.1
Round feedback								
0	94.83	93.77	1.06	0.69	75	3	96.7	119.1
1	93.75	95.06	-1.31	0.83	35	3	96.2	177.7
2	95.00	88.34	6.66	1.79	65	3	95.5	120.8
3	95.22	94.65	0.57	2.24	80	3	96.5	119.4
4	94.63	94.91	-0.28	1.51	80	3	96.1	117.6
Probe + round								
0	94.83	94.92	-0.09	0.47	85	3	147.4	148.0
1	93.75	95.30	-1.55	0.55	95	3	130.9	128.4
2	95.00	95.33	-0.33	1.17	85	3	153.5	127.9
3	95.22	95.64	-0.42	0.80	95	3	163.9	169.3
4	94.63	94.56	0.07	0.83	85	3	195.1	156.1

Table 21 separates observation, revision, local trials, and advancement. Inspect/edit/probe/search columns count successful calls; failed calls are counted separately. One search can contain multiple local evaluations, so search calls and probe-budget consumption are distinct quantities. The interval sequence sums to twenty rounds in every row. An edit at completed round t takes effect for subsequent execution, beginning at round $t + 1$. Create and save calls remain in the underlying ledgers but are outside these four research-operation columns.

Table 21: Feedback workflows: successful research calls, failed calls, and execution schedules.

Seed	Inspect	Edit	Probe	Search	Failed	Round intervals	Edit after t
Fixed program							
0	2	0	0	0	0	20	–
1	2	0	0	0	0	20	–
2	2	0	0	0	0	20	–
3	2	0	0	0	0	20	–
4	2	0	0	0	0	20	–
One-shot							
0	2	0	0	0	0	20	–
1	3	0	0	0	0	20	–
2	2	0	0	0	0	20	–
3	3	0	0	0	0	20	–
4	2	0	0	0	0	20	–
Round feedback							
0	5	1	0	0	0	1/1/18	1
1	5	1	0	0	0	1/1/18	1
2	5	1	0	0	0	1/1/18	1
3	5	1	0	0	0	1/1/18	1
4	5	1	0	0	0	1/1/18	1
Probe + round							
0	4	1	1	1	1	1/1/18	1
1	4	0	1	1	1	1/1/18	–
2	5	1	0	2	1	1/1/18	1
3	4	2	1	2	0	1/1/18	1,1
4	5	2	1	2	1	1/1/18	1,1

Figure 10 provides the paired training trajectories underlying the endpoint and AUC summaries. Each condition has a full row of five seeds. Shared axes make both early training and final accuracy directly comparable across panels; intervention boundaries locate recorded edits along the same round axis.

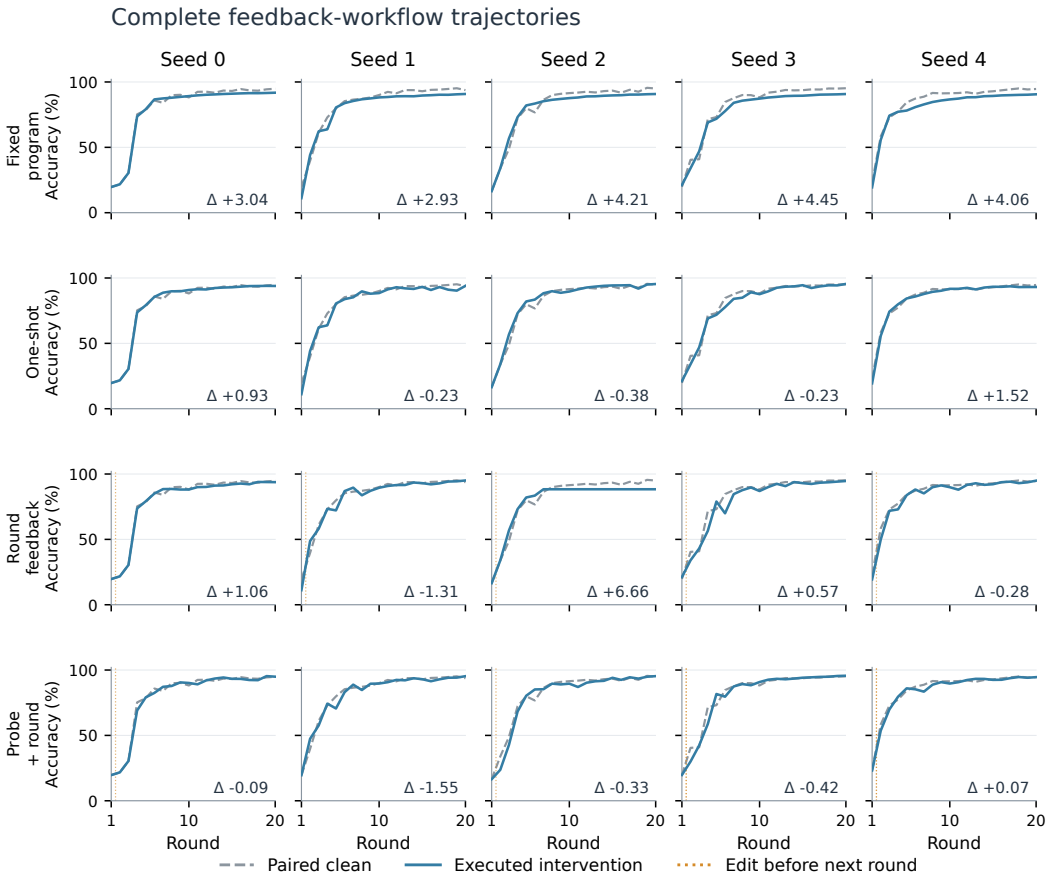

Figure 10: All twenty feedback-workflow trajectories. Gray dashed curves are same-seed clean controls; blue curves are the executed interventions. Gold dotted boundaries follow edits at completed rounds and precede the next round. Each panel reports its final clean-minus-attacked accuracy difference Δ in pp. All panels use the same accuracy and round scales.

L.2 Information views: every seed and observation schedule

Tables 22 and 23 expand the four authorized views under their common task. Alongside outcomes, they preserve each episode's advancement intervals and edit positions. Thus, the mean decision counts in the main text can be traced to the individual choices of when to obtain new federated feedback. The successful and failed call counts use the same convention as the feedback study.

Table 22: Information views: every evaluated seed. Accuracy and admission are percentages; drops are percentage points.

Seed	Clean	Attacked	Final drop	AUC drop	Admission	Decisions	Tokens (k)	Time (s)
Own messages								
0	94.83	94.92	-0.09	0.42	35	6	476.3	240.0
1	93.75	94.93	-1.18	1.05	90	4	297.3	183.6
2	95.00	95.58	-0.58	-0.87	0	5	369.2	208.2
3	95.22	93.39	1.83	2.19	75	5	297.9	184.0
4	94.63	94.38	0.25	-0.43	80	6	336.4	188.5
Population summaries								
0	94.83	94.52	0.31	-0.28	40	5	399.6	241.9
1	93.75	94.94	-1.19	0.79	25	5	286.7	162.9
2	95.00	95.18	-0.18	1.42	85	6	317.2	201.4
3	95.22	94.98	0.24	0.12	0	5	375.2	198.3
4	94.63	93.94	0.69	0.67	0	5	291.4	198.9
Population statistics								
0	94.83	95.71	-0.88	-2.28	95	3	142.6	105.1
1	93.75	95.30	-1.55	0.55	95	3	168.8	117.3
2	95.00	95.24	-0.24	0.66	85	6	259.6	144.3
3	95.22	95.07	0.15	2.26	85	4	232.8	134.3
4	94.63	94.45	0.18	0.03	95	3	172.5	120.1
Full messages								
0	94.83	97.02	-2.19	-3.56	95	3	197.5	147.5
1	93.75	94.68	-0.93	-1.74	90	5	279.0	141.2
2	95.00	96.48	-1.48	-2.82	95	3	148.8	104.9
3	95.22	96.06	-0.84	-1.89	95	4	236.7	182.9
4	94.63	94.47	0.16	-1.04	25	3	156.9	86.9

Table 23: Information views: successful research calls, failed calls, and execution schedules.

Seed	Inspect	Edit	Probe	Search	Failed	Round intervals	Edit after t
Own messages							
0	7	4	4	1	5	1/1/1/1/1/15	1,2,3,4
1	8	2	2	3	2	1/1/1/17	1,2
2	7	4	4	1	2	1/1/1/1/16	1,2,3,4
3	10	2	2	3	1	1/1/3/5/10	2,5
4	4	3	4	2	2	1/1/1/4/1/12	2,7,8
Population summaries							
0	3	4	4	2	3	1/1/1/1/16	1,2,3,4
1	4	2	3	2	2	1/1/1/1/16	2,3
2	4	2	2	1	3	1/1/1/5/6/6	1,2
3	7	3	3	1	3	1/1/1/5/12	1,2,3
4	6	3	0	1	3	1/1/1/1/16	1,2,3
Population statistics							
0	3	1	0	2	2	1/1/18	1
1	6	1	1	2	0	1/1/18	1
2	5	2	0	2	1	1/1/1/5/6/6	1,2
3	5	2	1	2	1	1/3/4/12	1,8
4	4	1	0	2	3	1/1/18	1

Seed	Inspect	Edit	Probe	Search	Failed	Round intervals	Edit after t
Full messages							
0	4	1	1	2	1	1/1/18	1
1	3	4	1	2	2	1/1/1/1/16	1,1,2,3
2	3	1	0	2	2	1/1/18	1
3	5	1	0	2	2	1/1/4/14	1
4	3	2	1	1	1	1/1/18	1,1

Figure 11 shows every clean/intervention pair. These trajectories complement the timing-and-cost view in Figure 7: the main figure locates decisions and resources, while the full curves show the training process observed under each resulting intervention schedule. The records distinguish local defensive influence, timing, resource use, and training outcome for the same episode.

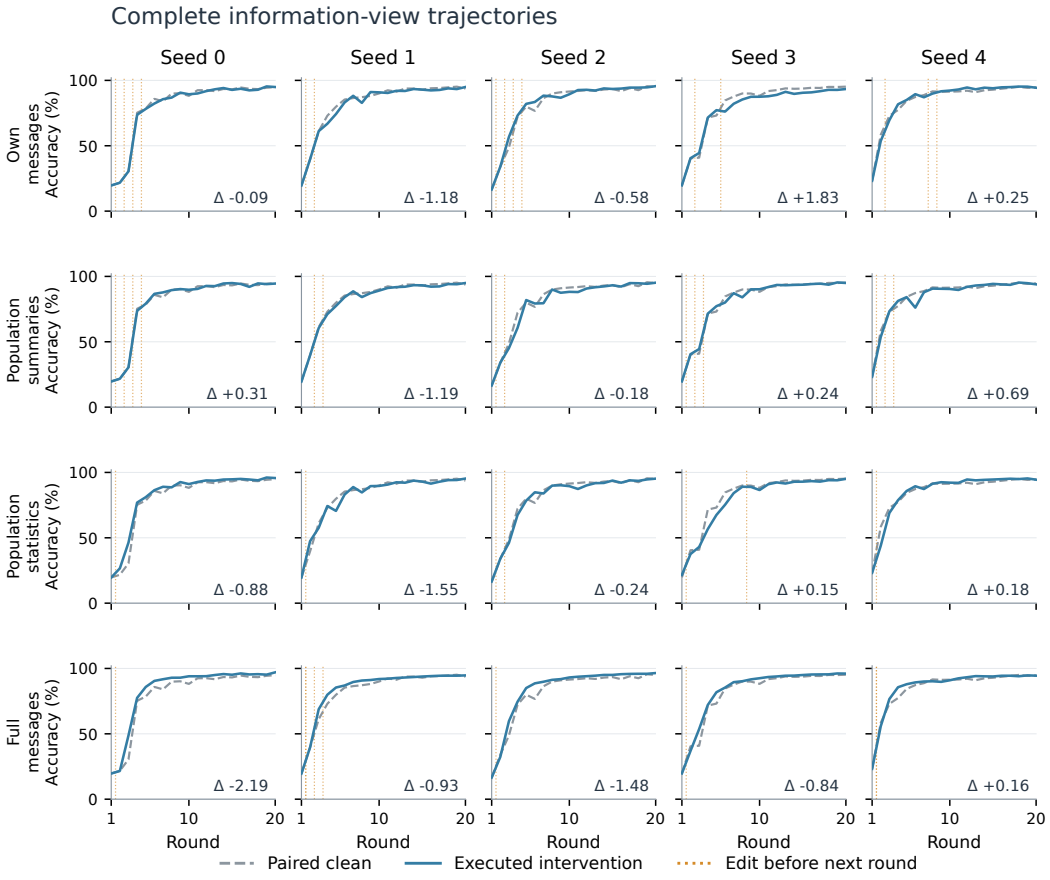

Figure 11: All twenty information-view trajectories, with the same encoding and shared scales as Figure 10. Each row contains five independent agent episodes under one authorized view, paired with their respective clean controls. Gold boundaries mark recorded edits; Δ is final accuracy drop in pp.

M Resource Use and Executed Search Comparisons

M.1 Cost accounting

Table 24 aggregates the forty episodes listed in Appendix L. Each condition has five episodes and each episode executes twenty federated rounds. Tokens count the recorded input-plus-output usage, including repeated context across agent turns. The scripted control has no LLM token cost. Elapsed time is the recorded candidate-episode duration, which includes its agent interaction and execution but excludes the separately run clean control. The measurements describe the archived execution environment; token counts are not monetary estimates or comparisons with human effort.

Table 24: Resources per recorded episode ($n = 5$ per condition, mean $\pm$ sample SD). Tokens are input plus output; time excludes the paired clean run. Each episode executes 20 rounds.

Condition	Tokens (k)	Time (s)	Advancements	Edits
Feedback workflows				
Fixed program	–	64.6 ± 0.2	1.0 ± 0.0	0.0 ± 0.0
One-shot	40.6 ± 3.0	99.7 ± 10.6	1.0 ± 0.0	0.0 ± 0.0
Round feedback	96.2 ± 0.4	130.9 ± 26.2	3.0 ± 0.0	1.0 ± 0.0
Probe + round	158.2 ± 23.9	145.9 ± 17.9	3.0 ± 0.0	1.2 ± 0.8
Information views				
Own messages	355.4 ± 73.9	200.8 ± 24.1	5.2 ± 0.8	3.0 ± 1.0
Population summaries	334.0 ± 50.8	200.7 ± 28.0	5.2 ± 0.4	2.8 ± 0.8
Population statistics	195.3 ± 48.8	124.2 ± 15.3	3.8 ± 1.3	1.4 ± 0.5
Full messages	203.8 ± 54.8	132.7 ± 37.7	3.6 ± 0.9	1.8 ± 1.3

The feedback workflows deliberately expose different opportunities to observe and revise. One-shot runs use one advancement; round-feedback and probe-plus-round runs use the prescribed 1/1/18 cadence. Information-view runs instead allow the agent to choose its advancement intervals. This difference is retained when interpreting resource use. Statistics-view episodes average 45.1% fewer tokens than own-message episodes in the recorded sample; the individual values and timings are available in the complete episode tables.

M.2 What the search counts measure

Across the five probe-plus-round episodes and twenty information-view episodes, successful search calls execute 219 local candidate evaluations: 144 explicitly supplied variants and 75 scalar-search settings. Explicit variants may change structure or parameters; the count describes the search API, not 144 distinct structural designs. Scalar searches comprise 11 norm-constraint multipliers, 58 direction coefficients, and six reference coefficients. Their populations are statistical in 95 evaluations, disclosed-message in 54, and coalition-local proxies in 70. These are evaluated candidates rather than unique algorithms or independent training runs. The forty-episode ledger separately records 37 successful direct probe calls. A search may evaluate several candidates against the same snapshot; repeated candidates and numerical refinements remain in the count because they consume trial budget.

The construction and scalar comparisons in Figure 4 exemplify this accounting: five structural variants and seven norm multipliers use twelve trial units at the same observed round. The later rejected probe contributes no successful evaluation. This ledger measures exploration actually performed under each view rather than a fraction of the full, program-defined design space.

N Component and Interface Reference

The reference below connects the formal search dimensions to their executable vocabulary. It enumerates forty-two registered primitives across ten execution surfaces, four dataset loaders, eighteen model aliases, and the twelve stable tool registrations. These are implementation inventories; the evaluated combinations are identified separately in Appendix A. Attack and defense recipes compose these primitives as described in Appendix G.

N.1 EXECUTION SURFACES AND AUTHORITY

The client surfaces are training configuration, batches, loss, gradients, optimizer steps, and uploaded updates. The server surfaces are evidence extraction, selection/weighting, sanitization, and reduction. Each primitive is validated for its surface, parameters, and required authority before becoming part of an installed program.

Table 25: Primitive inventory and surface-level control requirements. Parameter-dependent observation and sharing requirements are checked in addition.

Surface	Count	Required control	Execution location
train	1	malicious_local_training	Before local training
batch	9	malicious_batches	Input and label preparation
loss	2	malicious_local_training	Local objective
gradient	3	malicious_local_training	Before optimizer update
step	2	malicious_local_training	After optimizer update
update	9	malicious_updates	Uploaded message
evidence	5	server_aggregation	Message evidence
decision	4	server_aggregation	Retention and weights
sanitize	2	server_aggregation	Message transformation
reduce	5	server_aggregation	Global update reduction

Client-local parameters and returned tensors retain their declared permissions. A population reference, for example, is resolved under the observation contract; enabling a message-control surface does not grant observation of every client. A defense composition can also serve as a local trial rule. Installing it as the real server rule separately requires server-control authority.

Table 26: All 42 registered primitives, grouped by execution surface. Parameters list types, required fields, defaults, and bounds.

Primitive	Operation and parameter contract
train	
train.scale_hyperparameters	Scale local epochs, learning rate, and batch size for controlled clients. epochs_factor: number; default 1.0; min 1e-12 learning_rate_factor: number; default 1.0; min 1e-12 batch_size_factor: number; default 1.0; min 1e-12
batch	
input.gaussian_noise	Add deterministic Gaussian noise to selected inputs. std: number; required; min 0 mean: number; default 0.0 clamp_min: number / null; default null clamp_max: number / null; default null
input.partitioned_patch	Stamp the client-assigned part of a coalition-wide composite trigger. value: number; required height: integer; default 1; min 1 width: integer; default 4; min 1 grid_rows: integer; default 2; min 1 grid_columns: integer; default 2; min 1 row_gap: integer; default 2; min 0 column_gap: integer; default 2; min 0 row_offset: integer; default 0; min 0 column_offset: integer; default 0; min 0
input.patch	Stamp a constant rectangular trigger on selected samples. value: number; required height: integer; default 3; min 1 width: integer; default 3; min 1 location: string; default "bottom_right"; choices bottom_right, top_left, center, random

Continued from previous page

Primitive	Operation and parameter contract
input.scale	Scale selected input samples. factor: number; required
label.cyclic_shift	Cyclically shift selected labels within the class space. num_classes: integer; required; min 2 shift: integer; default 1
label.map	Map selected source labels to target labels. mapping: object; required; values: integer
sample.duplicate_selected	Append copies of selected samples to the local batch. copies: integer; default 1; min 1; max 16
sample.select_labels	Select samples whose current labels belong to a set. labels: array of integer; required combine: string; default "replace"; choices replace, intersect, union
sample.select_ratio	Select a deterministic fraction of the current local batch. ratio: number; required; min 0; max 1 mode: string; default "random"; choices random, first, last combine: string; default "replace"; choices replace, intersect, union
loss	
loss.parameter_distance	Add distance from the broadcast model as a differentiable stealth objective. coefficient: number; required; min 0.0 norm: string; default "l2"; choices l1, l2 squared: boolean; default false epsilon: number; default 1e-12; min 0.0
loss.scale_task	Scale the ordinary local task loss while preserving later regularizers. factor: number; required; min 0.0
gradient	
gradient.gaussian_noise	Add deterministic Gaussian noise to local gradients. std: number; required; min 0.0 mean: number; default 0.0
gradient.mask_magnitude	Keep only the largest- or smallest-magnitude gradient coordinates. ratio: number; required; min 0.0; max 1.0 keep: string; default "smallest"; choices smallest, largest per_layer: boolean; default false
gradient.scale	Scale local gradients before the optimizer step. factor: number; required
step	
step.project_l2	Project the local model into an L2 ball around the broadcast model. max_distance: number; required; min 0.0 epsilon: number; default 1e-12; min 0.0
step.project_linf	Project the local model into an L-infinity box around the broadcast model. max_distance: number; required; min 0.0
update	
update.clip_l2	Project the update into a global L2 ball. max_norm: number; required; min 0 epsilon: number; default 1e-12; min 0
update.clip_linf	Project every coordinate into an L-infinity box. max_abs: number; required; min 0
update.gaussian_noise	Add deterministic Gaussian noise to the update. std: number; required; min 0 mean: number; default 0.0 relative_to_norm: boolean; default false
update.history_blend	Blend with this attacker's previous emitted update. current_weight: number; default 0.5; min 0; max 1

Continued from previous page

Primitive	Operation and parameter contract
update.mean_direction	Replace or augment the update with a client-group mean direction. source: string; default "benign"; choices all, benign, malicious multiplier: number; default -1.0 mode: string; default "replace"; choices replace, add
update.negate	Reverse selected update layers and optionally scale them. factor: number; default 1.0; min 0
update.norm_match	Match the update norm to a statistic of a reference group. source: string; default "benign"; choices all, benign, malicious statistic: string; default "median"; choices mean, median, min, max multiplier: number; default 1.0; min 0 epsilon: number; default 1e-12; min 0
update.scale	Multiply selected update layers. factor: number; required
update.topk	Keep only the largest-magnitude update coordinates. ratio: number; required; min 0; max 1 per_layer: boolean; default false
evidence	
evidence.cluster_outlier	Score membership in the dominant direction cluster; outliers score one. cosine_threshold: number; default 0.5; min -1.0; max 1.0 epsilon: number; default 1e-09; min 0.0
evidence.cosine_similarity_max	Score each update by its highest cosine similarity to another client. accumulate: boolean; default true epsilon: number; default 1e-09; min 0.0
evidence.history_deviation	Score how far each client moved from its own previous update. epsilon: number; default 1e-09; min 0.0
evidence.krum_score	Score each update by the summed squared distance to its n-f-2 nearest neighbours, where n is the round size and f is assumed_byzantine; requires $n > 2f + 2$. assumed_byzantine: integer; required; min 0
evidence.l2_norm	Use each client update's global L2 norm as scalar evidence. No parameters.
decision	
decision.keep_below	Keep every update whose evidence score is at or below a threshold. threshold: number; required fallback_all: boolean; default true
decision.keep_lowest	Keep the requested number of clients with the lowest evidence scores. count: integer; required; min 1
decision.median_band	Keep evidence scores inside a multiplicative band around their median. lower: number; default 0.1; min 0.0 upper: number; default 3.0; min 0.0 fallback_all: boolean; default true
decision.reweight_foolsgold	Convert similarity evidence into continuous trust weights. epsilon: number; default 1e-06; min 0.0
sanitize	
sanitize.clip_l2	Clip selected updates to a fixed or median-derived L2 threshold. threshold: number; required; min 0.0 threshold_mode: string; default "fixed"; choices fixed, median epsilon: number; default 1e-12; min 0.0
sanitize.gaussian_noise	Add deterministic Gaussian noise to selected updates. std: number; required; min 0.0 mean: number; default 0.0

Continued from previous page

Primitive	Operation and parameter contract
reduce	
reduce.coordinate_median	Coordinate-wise median over selected updates. No parameters.
reduce.geometric_median	Smoothed Weiszfeld geometric median over selected updates. iterations: integer; default 5; min 1 epsilon: number; default 1e-06; min 0.0
reduce.mean	Arithmetic mean over selected sanitized updates. No parameters.
reduce.trimmed_mean	Coordinate-wise trimmed mean over selected updates. fraction: number; required; min 0.0; max 0.49
reduce.weighted_mean	Trust-weighted mean over positively weighted updates. epsilon: number; default 1e-12; min 0.0

N.2 Dataset loaders, model aliases, and packaged scenarios

The dataset registry contains `MNIST` (LeCun et al., 1998), `Fashion-MNIST` (Xiao et al., 2017), and `CIFAR-10`/`CIFAR-100` (Krizhevsky, 2009). Dataset transforms, partitioning, sample selection, and input resizing are scenario configuration fields. Model families include LeNet (LeCun et al., 1998) and ResNet (He et al., 2016), with Group Normalization in the named GN variants (Wu & He, 2018). Table 27 records all eighteen model aliases and the shape-dependent routing implemented by the model manager. Aliases sharing a constructor are grouped, so the alias count is distinct from a count of unique architectures.

Table 27: Registered model aliases and constructor routing. Shape-dependent dispatch is shown explicitly.

Aliases	Constructor / routing
lenet	LeNet; 32x32 inputs route to LeNetCIFAR
lenet5	LeNet5; requires 32x32 input
lenet_cifar	LeNetCIFAR
linear, logistic	LinearNet
cnn, simple_cnn	SimpleCNN; 28x28 inputs route to CNNMNIST
deep_cnn	DeepCNN
cnn_mnist	CNNMNIST
resnet18	ResNet18; 32x32 routes to ResNet18CIFAR, 28x28 to ResNet18MNIST
resnet34	ResNet34
resnet50	ResNet50
resnet18_cifar	ResNet18CIFAR
resnet18_gn, resnet18_cifar_gn	ResNet18CIFARGroupNorm
resnet20_cifar	ResNet20CIFAR
resnet34_cifar	ResNet34CIFAR
resnet18_mnist	ResNet18MNIST

Three packaged scenarios instantiate the configuration interface: `krum_escape`, `protocol_native_mnist`, and `protocol_native_cifar_krum`. They package a research question, federation configuration, permissions, and evaluation settings. The controlled interaction studies in this paper use the MNIST/LeNet-5 configuration; the historical L1 case uses its separately specified CIFAR-10/CNN world. The catalog supplies configuration choices, while each experiment specifies which combination it evaluates.

N.3 STABLE TOOL INPUTS AND SHARED RESEARCH OBJECTS

Table 28 includes every input field advertised by the twelve stable tool registrations. Table 29 factors their nested structures into reusable objects. The typed object schemas reject undeclared fields, while deliberately open maps such as metadata, primitive parameters, and JSON merge patches are interpreted by their corresponding validators. Defaults are shown only where explicitly advertised; omitted optional values can also be normalized by the object parser.

Tasks select a subset of these registrations. In particular, the evaluated L2 tasks expose inspection, attack construction/revision, local trials, advancement, and saving as configured in Appendix I. The three `experiment.*` registrations support separately authorized experiment orchestration; fork/restore is not exposed in these L2 tasks. Their real-round budgets and irreversible within-episode execution therefore retain the semantics in Section 4.4.

Table 28: The complete input field surface of the twelve stable tool registrations. Named nested objects are expanded in Table 29.

Tool / field	Input contract
attack.create	
spec	AttackProgram; required
attack.edit	
program_id	string; required
patch	object; required
attack.save	
program_id	string / null
name	string; required
hypothesis	string; required; min length 1
defense.create	
spec	DefenseProgram; required
defense.edit	
program_id	string; required
patch	object; required
experiment.compare	
checkpoint_ids	array of string; required; min items 2
experiment.fork	
name	string / null
persist	boolean; default false
experiment.restore	
checkpoint_id	string; required
federation.inspect	
resource	string; default "protocol"; choices protocol, state, geometry, attack_surface, defense_surface, installed_attacks, installed_defenses, methods
surface	string / null; choices batch, decision, evidence, gradient, loss, reduce, sanitize, step, train, update, null
include_unavailable	boolean; default false
method	string / null
federation.probe	
program_id	string / null
patch	object / null
surrogate	DefenseProgram or null
surrogate_recipe	string / null
surrogate_parameters	object
federation.search	
program_id	string / null
parameter	string / null
variants	array of Variant or null; minItems 1
bounds	array / null of number; min items 2; max items 2

Continued from previous page

Tool / field	Input contract
strategy	string; default "bisect"; choices bisect, scan
max_evaluations	integer; default 8; min 2
tolerance	number; default 0.0
min_weight_share	number; default 0.0
surrogate	DefenseProgram or null
surrogate_recipe	string / null
surrogate_parameters	object
federation.step	
rounds	integer; default 1; min 1

Table 29: Shared typed input objects. A field absent from the required list is optional at the advertised schema layer; parsing and capability checks further normalize or validate its value.

Object / field	Input contract
AttackProgram	
schema_version	string; choices 1.0
program_id	string; min length 1
name	string; min length 1
target_client_ids	array of integer (item min 0)
coalition	Coalition
train_pipeline	array of Atom; primitive names from the train surface
batch_pipeline	array of Atom; primitive names from the batch surface
loss_pipeline	array of Atom; primitive names from the loss surface
gradient_pipeline	array of Atom; primitive names from the gradient surface
step_pipeline	array of Atom; primitive names from the step surface
message	Message or null
update_pipeline	array of Atom; primitive names from the update surface
schedule	Schedule
seed	integer
enabled	boolean
metadata	object
DefenseProgram	
schema_version	string; choices 1.0
program_id	string; min length 1
name	string; min length 1
pipeline	array of Atom; minItems 1; server surfaces only
schedule	Schedule
seed	integer
enabled	boolean
metadata	object
Message	
reference	Vector
directions	array of Vector
support	Support
constraints	array of Constraint
Coalition	
member_ids	array of integer (item min 0)
roles	object; values: string; member-ID keys, string labels
leader_id	integer / null; min 0
sharing	string; choices isolated, coalition, population
emission	string; choices independent, replicate_leader, perturbed_leader
emission_noise	number; min 0
Schedule	
start_round	integer; min 1
end_round	integer / null; min 1

Continued from previous page

Object / field	Input contract
every	integer; min 1
probability	number; min 0; max 1
Vector	
source	string; choices zero, current, population, client, gaussian
group	string; choices own, coalition, benign, all
statistic	string; choices mean, median, std
client_id	integer / null; min 0
transform	string; choices identity, sign, unit_l2
coefficient	number
noise_mean	number; default 0.0
noise_std	number; default 1.0; min 0
Support	
layers	array of string; min items 1
topk_ratio	number / null; min 0; max 1
keep	string; choices largest, smallest
Constraint	
kind	string; required; choices clip_l2, clip_linf, norm_match, distance_ envelope, align_direction, history_smooth
value	number / null
center	string; choices zero, current, reference
source	string; choices own, coalition, benign, all
statistic	string; choices mean, median, min, max
multiplier	number; min 0
metric	string; choices max_l2, sum_l2_squared
iterations	integer; min 1
epsilon	number; min 0
Atom	
primitive	string; required; min length 1
params	object
layers	array of string; min items 1
enabled	boolean
Variant	
label	string / null
patch	object; required

Composition semantics. A message is a reference plus its direction terms in update-delta coordinates. Support selection and the ordered constraint list act on the result. Constraint kinds select the fields they consume: clipping uses radius and center; norm matching uses a reference statistic and multiplier; distance envelopes use a metric and search iterations; alignment uses a minimum cosine; history smoothing bounds motion from the client's previous emitted message. Coalition emission then specifies independent messages, leader replication, or perturbed replication. The schedule specifies the eligible rounds and activation probability. Local-training pipelines expose their additional client-side surfaces under the corresponding control permissions.

Editing and local search. An edit applies JSON Merge Patch to the installed specification, validates the resulting object, and records before/after fingerprints and the future effective round. A probe's patch is hypothetical. Search accepts either a continuous parameter path with two bounds or a list of labeled structural patches. The modes are mutually exclusive; each local candidate evaluation consumes one trial unit. Scalar search exposes bisection or scanning, evaluation limits, tolerance, and a minimum coalition-weight criterion. Bisection requires a bracket; scanning records the sampled response shape. All candidates are validated before their applicable trial cost is spent.

N.4 RETURN STRUCTURES AND EVIDENCE ATTRIBUTION

Every registry call returns a common envelope: a successful call has `ok=true` and a `result` object; a failed call has `ok=false` and an `error` object with type and message. Program-validation errors additionally return their validation report. Table 30 enumerates result fields, including conditional fields and the resource-specific inspection branches. Auxiliary prose fields explain an observation or operation; the structured identifiers and receipts carry its attribution.

Table 30: Return-field inventory of the stable tools. Fields depend on the selected resource, authorization, and operation branch; their conditions are explained below.

Tool / resource	Fields inside the successful result
attack.create	executions, history_clients, installed, receipt, spec, spec_fingerprint
attack.edit	executions, history_clients, installed, receipt, spec, spec_fingerprint
attack.save	hypothesis, name, note, program_id, receipt, saved_at, scored, spec_fingerprint, version
defense.create	authority, executions, installed, receipt, spec, spec_fingerprint
defense.edit	authority, executions, installed, receipt, spec, spec_fingerprint
experiment.compare	baseline_checkpoint_id, branches, receipt
experiment.fork	checkpoint_id, created_at, path, receipt, round_num, scenario_fingerprint, schema_version
experiment.restore	checkpoint_id, observation, receipt
federation.inspect / attack_surface	algebra, control_surfaces, message_views, primitives, principle, program_contract, recipes, unavailable_primitives, unavailable_recipes
federation.inspect / defense_surface	algebra, primitives, principle, program_contract, recipes, server_aggregation_control, unavailable_primitives, unavailable_recipes
federation.inspect / geometry	as_uploaded_l2_norms, available, last_aggregation_feedback, note, observed_round, pairwise, population_model, pre_intervention_l2_norms, reason, round_size, surrogate_fidelity, unseen_client_count, visible_client_ids
federation.inspect / installed_attacks	programs
federation.inspect / installed_defenses	defenses
federation.inspect / methods	methods, name, note, root, schema_version, versions
federation.inspect / protocol	agent_research_cycle, budgets, current_observation, dataset, defense_abstraction, federated, federated_algebra, feedback_cost_model, fingerprint, max_rounds, metric_providers, model, model_poisoning_abstraction, name, objective, schema_version, seed, server_policy, threat_model
federation.inspect / state	decision_budget, installed_attack_ids, observation
federation.probe	assumptions, budget, candidate_client_ids, candidate_groups, decision_boundary, decision_boundary_source, entries, evidence_kind, hypothetical, observed_round, population, predicted_admitted, predicted_rejected_client_ids, predicted_selected_client_ids, predicted_weight_share, probed_round, program, receipt, schema_version, surrogate
federation.search	boundary_interval, bounds, budget, budget_note, crossings, evaluations, evaluations_used, feasible_value, infeasible_value, min_weight_share, parameter, population, program_id, receipt, schema_version, scored_for_fidelity, status, strategy, surrogate
federation.step	info, observation, per_round, receipt, reward, rounds_executed, rounds_requested, surrogate_feedback, terminated, truncated

Continued from previous page

Tool / resource	Fields inside the successful result

Inspection. Protocol inspection returns the scenario, threat contract, budgets, policy, and current observation. State inspection returns an observation plus installed attack identifiers; a configured advancement budget adds its used/remaining counts and remaining rounds. Geometry is either unavailable with a reason, or contains its observed round, authorized client identifiers, honest-update norms, pairwise distances/cosines, and population coverage. Uploaded norms, aggregation feedback, and fidelity appear when disclosed and available. Surface inspection returns available primitives/recipes and missing capabilities. Method inspection returns a ledger or the versions of a named method; installed-program inspection returns program descriptions.

Program versions and saved methods. Attack descriptions contain the normalized specification, fingerprint, installation flag, executions, and history-client identifiers. Defense descriptions additionally identify agent or harness authority. Create/edit receipts identify the operation, program, new fingerprint, and effective round; edits also identify the prior fingerprint. A saved-method result identifies name, version, hypothesis, save time, and behavior fingerprint, with `scored=false` at save time. The harness later associates verified evaluation outcomes with the version that actually executed.

Local prediction and search. A probe records observed/probed rounds, surrogate identity, population representation and simulated/real sizes, candidate groups, program fingerprints, predicted selection and weight, and the decision boundary. Each `entries` element contains index, client identifier, role and role note, evidence score, rank, weight, selection, and margin. Its budget and receipt identify the trial cost and unchanged real-round state. Search returns strategy, status, bounds, feasible/infeasible values, boundary interval, crossing count, and evaluated candidates, together with its program, parameter, surrogate, population, budget, and receipt. An evaluation entry contains value, feasibility, weight share, selected identifiers, and margin. A reduced scalar-search budget can add `budget_note`. Search candidates are recorded with `scored_for_fidelity=false`; a committed direct prediction has separate attribution.

Real advancement and experiment orchestration. Step returns the last observation, reward, termination/truncation flags, disclosed information, and a receipt linking initial/final round and installed programs. Batched advancement additionally returns requested/executed counts and per-round metrics, rewards, and information; disclosed surrogate feedback is conditional on a comparable prediction. Fork returns checkpoint identity, scenario fingerprint, round, time, and receipt, adding a path when persistence is requested. Restore returns the restored observation and checkpoint identity. Compare returns the baseline checkpoint and branches, each containing round, final metrics, objective score, attack/defense specifications, and differences from the first branch. These experiment-management records are separate from the agent-visible L2 action set.

O Controlled Record-Replay Ablations

Design and unit of analysis. We take the five probe-plus-round episodes and five episodes from each of the own-message, population-summary, and population-statistics views. All twenty traces include local trials and lack full-message authority. For each original trace, four separate copies receive one predefined semantic mutation. We preserve the original records on disk and replay their copies through the same verifier. The task's declared client set, information permissions, round horizon, objective, and fixed defense determine the verification context.

Mutations. Client-authority mutations add a benign client to one recorded intervention, keeping the modified-client set a subset of the audited set. Trial-clock mutations change one trial receipt to report real advancement. Population-authority mutations change one trial's population label to disclosed real messages, which these views do not authorize. Lifecycle mutations remove one after-aggregation event, then restore contiguous record indices; generic index continuity therefore remains satisfied.

Each mutation represents a known violation of the declared experimental contract, not an observed failure in the original episode.

Ablation. The full verifier and each ablated condition receive identical records. An ablation suppresses the designated error-code family while retaining all other checks. This isolates the contribution of the four implemented check families; it is not a comparison against another agent framework or a test of unrestricted fault coverage. We do not mutate cryptographic digests to represent an adversarial trace-forgery experiment.

All twenty original traces are accepted. The full verifier rejects all eighty injected variants; omitting one corresponding check family misses those twenty variants and retains detection of the other sixty. Removing all four leaves all eighty undetected by the remaining checks. Counts describe these predefined perturbations, rather than a naturally occurring error distribution. The CPU interface tests in Appendix K separately test state preservation, installation, and live permission filtering.

P From Local Predictions to Executed Constructions

P.1 Matching predictions to subsequent execution

The forty archived episodes contain 800 real rounds and 256 successful local candidate evaluations: 219 search evaluations and 37 direct probes. We join trial records to the subsequent upload only when program identity and behavioral fingerprint match, the trial uses the immediately preceding completed-round snapshot, and its record precedes execution. The fingerprint excludes program names and descriptive metadata but retains all behavioral fields. When several predictions match the same upload, we use the latest one. Unexecuted candidates and revisions without a matching trial contribute no comparison.

This rule yields 44 next-round comparisons from 24 episodes. The target is whether a controlled client is selected in the next real round; every setting here uses single-selection Krum. We additionally recover each trial's rule parameters from its tool call: all 28 coalition-proxy comparisons use agent-specified Byzantine count $b = 0$, whereas the real ten-client server uses $b = 2$. The statistical and disclosed-message comparisons retain $b = 2$. All use one selected update. Thus, coalition-proxy disagreement combines population and local-rule differences with changes in training state.

Agreement measures *transfer of a selected candidate's local prediction*, not calibration on identical tensors or isolated measurement of one mismatch source. Counts within an episode are repeated observations, not independent experimental replications.

The coalition-proxy group contains 28 comparisons from ten episodes: all predict controlled-client selection, six are followed by real selection, and 22 by rejection. The statistical group contains eleven comparisons from ten episodes: nine predict selection, seven of those are selected, and both predicted rejections agree with execution. The disclosed-message group contains five comparisons from four episodes: all predict selection and four are selected. Agreement is therefore 6/28, 9/11, and 4/5, respectively.

Selection and modification are recorded separately. One of the six controlled-client selections in the coalition-proxy group uploads an unchanged message; the counts of selected *modified* messages are five, seven, and four in the respective groups. The main outcome tables use modified-message admission. The comparison above instead matches the probe's prediction of candidate selection, preserving the distinction between controlling a selected client and changing its update.

The population label distinguishes a local judgment on coalition proxies from a judgment on the full fresh federation. These descriptive groups come from their respective authorized workflows; they do not isolate population representation from the policies selecting candidates.

P.2 Classifying the final executed expression

We recover each installed specification from its successful creation or edit record and join it to upload audits using the full specification fingerprint. For each episode, we combine identical symbolic terms and classify the final executed expression before constraints: identity-transformed current/population terms with positive coefficients; negative coefficients only; mixed coefficient signs; a zero expression;

or transformed/other terms. Zero-valued terms are ignored. This is a syntactic description, not a measured cosine similarity, a proof that a construction is benign, or a complete description of later constraints and coalition emission.

All five probe-plus-round episodes end with positive current/coalition terms. In the round-feedback workflow, four end in this category and one uses a zero expression. One-shot episodes comprise three positive-term and two negative-term expressions. In the information-view study, own-message episodes all use positive terms; population-summary episodes use four positive-term and one negative-term expression; statistics-view episodes use four positive-term and one mixed-sign expression; full-message episodes use two positive-term expressions and three containing transformed terms.

The zero-expression episode is the documented zero-message revision; its final drop is 6.66 pp, while the other four round-feedback episodes average approximately 0.01 pp. Reporting all five preserves the distinction between the illustrated intervention and typical group behavior. The fixed-program group remains the full-workflow reference in Figure 6; its exact message and coalition settings are specified in Appendix A.

The five probe-plus-round programs all use replicated emission and norm matching; three have norm matching alone, one additionally smooths history, and one additionally imposes a distance envelope. Their norm multipliers are 1, 1, 0.867, 2, and 1. In contrast, the fixed reference attenuates the coalition mean by a factor of 0.1 without norm matching. These recorded constructions make the magnitude and constraint choices explicit alongside direction; the syntax categories alone do not determine attack effectiveness.

Q Isolating Coalition Geometry in the Existing Component Study

Checking the intervention itself. We compare the compiled objects retained with the original seven-attack, three-defense, five-seed matrix. After excluding program names, identifiers, and descriptive metadata, the replicated and perturbed configurations differ only in coalition emission and its perturbation amplitude. Message references, directions, constraints, schedules, controlled clients, and training settings are unchanged. This is a researcher-configured component comparison, not an agent-generated attack discovery.

What the component changes. For a nonzero leader message v, the perturbed emitter draws a client-specific unit vector e_i orthogonal to v and returns

$$u_i = \frac{v + \rho\|v\|_2 e_i}{\sqrt{1+\rho^2}}, \qquad \|u_i\|_2 = \|v\|_2, \qquad \cos(u_i, u_j) = \frac{1+\rho^2\langle e_i, e_j\rangle}{1+\rho^2}. \tag{12}$$

The implementation renormalizes the constructed message to the leader's norm; zero-norm cases return the original message. The equation shows the intended finite-input operation: changing joint angular geometry need not change individual message norms. The experiment uses $\rho = 1$. Similarity-based defenses and distance-based selection can therefore respond differently to the same emission change, as the recorded results demonstrate.

Paired outcomes. Subtracting the replicated attack's accuracy drop from the perturbed attack's drop gives FoolsGold differences of $85.99, 86.30, 86.27, 78.78, 84.43$ pp over the five seeds. The mean increment is 84.35 pp, with a descriptive paired-t 95% interval $[80.37, 88.34]$ pp. The corresponding differences are zero for every Krum seed and $-0.05, -0.17, -0.24, -0.05, -0.07$ pp for trimmed mean. These comparisons concern this fixed construction and training horizon.

All original observations remain in the reported matrix. As already marked in Table 7, one perturbed/FoolsGold run has non-finite loss but a recorded accuracy trajectory. A sensitivity summary over the four finite-loss pairs gives an 83.87-pp mean increment. Thus the defense-specific contrast is also present outside that numerical failure; the sensitivity summary does not replace the original five-seed estimate.

R WHY A COALITION PROXY CAN GIVE UNINFORMATIVE FEEDBACK

R.1 A STRUCTURAL CONSEQUENCE OF THE LOCAL POPULATION

Let $d(x, y) = \|x - y\|_2^2$. Krum assigns a message the sum of distances to its $q = n - b - 2$ nearest *other* messages and selects a minimum-score message. This is the self-excluding convention used by the executor in our experiments.

Proposition (replica dominance in a four-message proxy). For the proxy population (h_1, h_2, v, v) with $b = 0$, each replica has score

$$s(v) = \min\{d(v, h_1), d(v, h_2)\}. \tag{13}$$

Neither honest proxy has a smaller score. If h_1, h_2, v are pairwise distinct, a replica is selected regardless of the direction or magnitude of v.

Proof. Here $q = 2$. A replica's neighbors include its identical twin at distance zero and the nearer honest proxy. Writing $a = d(h_1, h_2)$ and $d_i = d(v, h_i)$, the score of h_i is the sum of the two smallest elements of (a, d_i, d_i), namely $d_i + \min(a, d_i)$. This is at least d_i and hence at least $\min(d_1, d_2)$. Pairwise distinctness makes the first inequality strict. Coincident vectors can tie; the score statement still holds. □

Why the real decision can be different. In a ten-client round with $b = 2$ and two replicas, $q = 6$: each replica must include distances to at least five honest messages. If honest messages have norm at most R, its score is at least $5(\|v\|_2 - R)^2$ for $\|v\|_2 > R$. An honest message can choose six of the other seven honest messages, giving score at most $24R^2$. Thus sufficiently large replicas are rejected by the full population even though the four-message proxy keeps selecting one. The local aggregator executes correctly in both cases; it is the assumed population and neighborhood that change the meaning of its feedback.

This proposition identifies a specific, checkable failure mode rather than a claim that partial information always produces bad predictions. It explains why optimizing only the proxy's binary selection feedback can leave an agent with no discrimination along its magnitude-search direction. Retaining the candidate, population construction, and defender parameters makes that failure visible to the researcher.

R.2 A PAIRED SAME-SNAPSHOT INTERVENTION

We hold both the candidate tensor and the full-population reference fixed, and change only the construction and parameters of the local trial. Synthetic worlds use twenty random seeds, population sizes $7, 10, 21$, and dimensions $8, 64, 1024$. A separate real-data stratum uses MNIST/LeNet training seeds 100–104, with snapshots at rounds $1, 2, 5, 10, 20$ under the original twenty-round training configuration. No agent chooses the candidates in this study.

For each snapshot we replicate $v = \gamma \overline{u}_{\mathcal{C}}$ for two controlled clients, with $\gamma \in \{0, \pm 0.1, \pm 0.25, \pm 0.5, \pm 1, \pm 2, \pm 4, \pm 8, \pm 16\}$. The full-population reference replaces those clients' messages and evaluates Krum with $b = 2$ on that same snapshot. All distance recomputations use double precision in both the reference and trial. This removes future-round message drift and candidate-selection differences from the comparison.

The small coalition proxy appends the two replicas to the two original coalition messages ($n = 4, b = 0$). A size-preserving proxy repeats those same authorized originals until it has $n - 2$ proxy rows and appends the replicas; it is evaluated with $b = 0$ and $b = 2$. This control uses no additional client information. Statistical populations draw $n - 2$ rows from the disclosed coordinate means and variances, using four independent draws shared across candidate magnitudes within a snapshot. Disclosed-message controls use actual rows, with both $b = 0$ and $b = 2$. The $n = 4, b = 2$ combination is invalid under the executor's $n > 2b + 2$ requirement and is not substituted for a runnable Krum rule.

Results and unit of analysis. All 400 direct numerical checks satisfy Equation 13. In the trained-state stratum, the small proxy selects a replica for all 425 candidates; full-population Krum rejects

334 of them. Holding $b = 0$ and only expanding the proxy population changes agreement from $91/425$ to $378/425$. Matching both population size and $b = 2$ gives $359/425$ agreement: it removes these false acceptances but misses 66 of the 91 truly accepted candidates. Thus repetition alone yields a conservative surrogate, not a recovered true population. Statistical trials agree in $1677/1700$ comparisons, with 23 false rejections and no false acceptances in these snapshots. The disclosed-message control agrees exactly; changing only its b to zero changes four decisions.

The generated-world stratum shows the same uninformative small-proxy behavior: all 3,060 predictions are positive, including 2,219 false acceptances. Size-preserving and statistical populations agree in $2304/3060$ and $11289/12240$ comparisons, respectively. All counts include the entire prescribed candidate grid. Candidates, snapshots, and statistical draws within a training seed are repeated measurements; the real-data replication count is five training seeds, not 1,700 independent experiments. These are same-snapshot surrogate checks, distinct from the next-round transfer measurements in Appendix P.